View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Repositório Institucional do UniCEUB

CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB
CURSO DE ENGENHARIA DE COMPUTAÇÃO
PROJETO FINAL

SISTEMA PARA ATENDIMENTO DE MESAS EM RESTAURANTES CONTROLADAS POR RADIOFREQUÊNCIA

Autor: Aline de Oliveira Ataide(RA: 20706600)

Orientador: Prof. Francisco Javier de Obaldia Diaz

BRASÍLIA-DF

2ºSEMESTRE DE 2011

ALINE DE OLIVEIRA ATAIDE

SISTEMA PARA ATENDIMENTO DE MESAS EM RESTAURANTES CONTROLADAS POR RADIOFREQUÊNCIA

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof.Msc. Francisco Javier De Obaldía Diaz

Brasília

Novembro,2011

ALINE DE OLIVEIRA ATAIDE

SISTEMA PARA ATENDIMENTO DE MESAS EM RESTAURANTES CONTROLADAS POR RADIO FREQUENCIA

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof. Msc.Francisco Javier De Obaldía

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas -FATECS.

_____

Prof. Abiezer Amarilia Fernandez

Coordenador do Curso

Banca Examinadora:

_____

Prof. Msc.Francisco Javier De Obaldía

Orientador

_____

Prof.Dr. Miguel Arcanjo Bacellar Goes Telles Junior

Instituição

_____

Prof. nome, titulação.

Instituição

_____

Prof. nome, titulação.

Instituição

Dedico esta Monografia aos meus pais que desde a minha infância têm dado grande incentivo ao meu desenvolvimento intelectual. Sem vocês eu não teria compreendido a importância do SABER.

AGRADECIMENTOS

Primeiramente a Deus pela saúde, pelas oportunidades e pelos desafios vencidos ou não, que possibilitam o meu engrandecimento e maturidade.

Aos meus Pais, José Sabino de Ataide e Cleusa de Oliveira Ataide, pelo carinho e preocupação que sempre tiveram. Pela educação que me proporcionaram, e, claro, pela vontade e dedicação incansáveis de me guiar e orientar a seguir, sempre, o melhor caminho.

Ao meu orientador Francisco Javier pelas valiosas orientações, paciência e interesse constante em ajudar-me.

A todos os professores do curso de Engenharia de Computação que passaram com paciência e profissionalismo a base do conhecimento deste projeto. Agradecimento especial ao professor Luís Cláudio, seu amor e dedicação por sua profissão nos motiva a ser cada vez melhores.

Aos grandes amigos Felipe Souto, José Carlos, Thiago Meneses que me deram todo o apoio e incentivo para a realização deste projeto e a todos aqueles que direta ou indiretamente, estiveram presentes e dispostos a ajudar de alguma forma na conclusão deste projeto.

A todos os colegas de sala que fizeram desses 5(cinco) anos de curso um percurso de vida mais alegre e engraçado.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS e ABREVIATURAS

A/D - Analógico / Digital

BIT - Binary Digit

CI - Circuito Integrado

CPU - Central Processing Unit (Unidade Central de Processamento)

GND - Ground (Terra)

GUI- *Graphical User Interface*

EIA- Eletronics Industries Association

IBM- International Business Machines

LASER - Light Amplification by Stimulated Emission of Radiation (Amplificação da Luz por Emissão Estimulada de Radiação)

LED - Light-Emitting Diode (Diodo Emissor de Luz)

MHz- Mega Hertz(10^6 Hertz)- Unidade de Frequência

PC - Personal Computer (Computador Pessoal)

pF-Pico Farad

PWM- Pulse Width Modulation

RAM - Random Access Memory (Memória de Acesso Aleatório)

RS-232 - Recommended Standard 232 (Padrão Recomendado Número 232)

Rx (RXD) - Receiving Signal (Sinal Recebido)

TI-Tecnologia da Informação

Tx (TXD) - Transmitting Signal (Sinal Transmitido)

USB- Universal Serial Bus

VCC - Positive Supply Voltage (Tensão de Alimentação Positiva)

RESUMO

Este estudo propõe o desenvolvimento de um sistema composto de módulos cujo funcionamento substitui os arcaicos meios de comunicação entre clientes de bares, restaurantes e similares e os garçons ou atendentes. O sistema consiste em um botão acessível pelo cliente que por meio de um microcontrolador PIC16F628A comunica via transmissão de radiofrequência o número da mesa que está solicitando atendimento a um receptor RF conectado a outro PIC16F628A que, por sua vez encaminha as informações para o conversor MAX232 para que o número da mesa possa ser exibido em realce em uma tela de um microcomputador.

Nesse sentido, temos como objetivos específicos a construção de um protótipo composto por 3 (três) módulos: 2 (dois) *hardware*s transmissores das informações por radiofrequência e 1(um) *hardware* receptor. Além disso, temos como objetivo construir uma interface gráfica para simular a organização das mesas do ambiente, gerenciar o sistema e mostrar o número de cada mesa em um monitor . O resultado do projeto atendeu aos objetivos propostos, mostrou que o sistema faz uso de equipamentos computacionais simples , de baixo custo e com materiais de fácil obtenção no mercado e facilitando por isso sua possível utilização comercial.

Palavras-chave: Atendimento do garçom, transmissão RF, microcontrolador PIC 16F628A, conversor MAX232.

ABSTRACT

This study proposes the development of a system composed of modules whose operation replaces the archaic means of communication between customers of bars, restaurants and similar and the waiters or attendants. The system consists in accessible button customer through a PIC16F628A microcontroller communicates via radio waves(RW) the number of desk requesting assistance to an RW receiver PIC16F628A connected to another which in turn forwards the information to the converter MAX232 so that the number of the table can be displayed in bold on a screen of a microcomputer.

In this sense, we have specific objectives to build a prototype consisting of 3 (three) modules: 2 (two) *hardware* transmitters of information and a radio frequency 1(one) receiving *hardware*. Furthermore, we aim to build a graphical interface to simulate the organization of the tables of the environment, manage the system and showthe number each table on a monitor. The result of the project meet the proposed objectives, showed that the system makes use of computer equipment simple, low cost and with materials readily available in the market and thus facilitating their possible commercial use.

Keywords: waiter service, microcontroller, radio waves, PIC

CAPÍTULO 1- INTRODUÇÃO

1.1-Apresentação do Problema

Os sujeitos envolvidos neste projeto são basicamente os clientes do restaurante e os garçons, e como resultado desta relação é espera-se que o garçom garanta rápido atendimento, que não falte nada aos clientes e que eles estejam satisfeitos com os produtos e serviços oferecidos pelo estabelecimento. E para isso, o primeiro passo, é que o cliente consiga chamar a atenção do garçom, normalmente ficando com o dedo levantado ou assobiando. Entretanto, essa realidade está ultrapassada pois o cliente pode perder muito tempo tentando chamar o garçom ou pode até mesmo não ser visto.

Todos esses tópicos apresentados nos fazem pensar na seguinte pergunta:

Como os clientes poderiam chamar o garçom de forma discreta e seguros de que seriam vistos?

1.2 – Objetivos

Este projeto tem como objetivo geral desenvolver um sistema para a automatização do atendimento de mesas em restaurantes e afins. O sistema irá permitir aos clientes solicitarem o atendimento do garçom por meio de um simples apertar de um botão.

Os objetivos específicos são:

- Construir dois *hardware*s transmissores das informações por radiofrequência;
- Implementar um *hardware* capaz de receber o número da mesa que está solicitando o pedido;
- Construir uma interface gráfica para simular a organização das mesas do ambiente, gerenciar o sistema e mostrar o número de cada mesa em um monitor;

1.3 – Justificativa e Importância do Trabalho

O maior objetivo das empresas é trabalhar para os clientes de forma a satisfazer suas necessidades, superar suas expectativas e desenvolver relações a longo prazo.

Tendo em vista a realização da Copa do Mundo em 2014 e das Olimpíadas em 2016, dois dos maiores eventos esportivos internacionais em solo brasileiro, um alto investimento é exigido dos Governos Federal, Estadual e Municipal junto com a iniciativa privada e patrocinadores.

Esses eventos prometem aquecer a economia nacional, e os principais reflexos devem ser sentidos nos Estados que vão receber os jogos da copa, suas capitais e também as cidades adjacentes. Um dos setores onde haverá maior repercussão é o setor de alimentação, por isso, é importante desde já pensar maneiras de agilizar o atendimento e melhorar os serviços prestados a fim de que o retorno dos investimentos seja garantido e que os clientes que estarão apenas visitando o Brasil voltem mais vezes e se tornem clientes fidelizados.

> Se você se limitar a fazer o bem básico, seu cliente irá considerá-lo ruim, a menos, é claro que nenhum concorrente chegue mais alto na hierarquia. Se você fizer bem o básico e o esperado, seu cliente irá considerá-lo medíocre, isto é, apenas satisfatório, mas sem nada especial. Se você fizer bem o básico, o esperado e o desejado, o cliente começará a favorecê-lo se você fizer tudo isso melhor que os concorrentes. E se você surpreender o cliente com o inesperado, terá um lugar especial em seu coração. Você terá a oportunidade de uma posição competitiva destacada.(ALBRECHT ,1995)

Essa citação por si justifica os investimentos em tecnologias da informação para os estabelecimentos de alimentação, uma vez que tais tecnologias podem surpreender os clientes tornando realidade os sonhos e desejos do consumidor. Quanto mais um empresário puder fazer para acrescentar valor aos produtos e serviços que oferece, mais os clientes voltarão e mais recomendações farão a respeito dele para colegas e amigos.

1.4 – Escopo do Trabalho

O escopo do projeto consiste em disponibilizar um protótipo de um sistema capaz de chamar um garçom até a mesa do cliente. A este projeto daremos o nome de Garçom Digital. O sistema é composto por dois *hardware*s transmissores que em um ambiente real ficariam dispostos nas mesas do restaurante, um *hardware* receptor que irá receber a informação com o número da mesa do cliente e irá passá-la para um monitor de computador e por fim, também compõe o sistema, uma aplicação que serve de interface entre o usuário e o módulo receptor.

O *hardware* receptor foi montado utilizando um microcontrolador da empresa Microchip, família PIC e modelo 16F628A. Neste microcontrolador são ligados o receptor de rádio frequência, utilizando a frequência de 433MHz, e um circuito integrado MAX232 que permite a comunicação no padrão RS-232 (EIA, 1969) com o computador.

Para o dispositivo de transmissão foi utilizado um botão simples e um microcontrolador igual ao do receptor . Neste microcontrolador está ligado o transmissor de rádio frequência, utilizando a mesma frequência do receptor (433MHz).

A interface do usuário foi desenvolvida utilizando a linguagem de programação Java, é executada na plataforma *Windows* e é a responsável pelo gerenciamento e simulação da organização das mesas do ambiente.

O projeto não contempla fazer pedidos discriminados por meio do *hardware* transmissor .

1.5 – Resultados Esperados

Como resultado do desenvolvimento deste projeto espera-se que seja possível com o apertar de um botão que o número correspondente à mesa apareça, em realce, na tela de um monitor.

1.6 – Estrutura do Trabalho

Este trabalho encontra-se divido em 6 capítulos, incluindo a INTRODUÇÃO, que trata de forma geral do tema proposto, os objetivos, justificativa e resultados esperados. O capítulo traz ainda esta seção, que descreve a estrutura e organização da monografia.

O segundo capítulo, APRESENTAÇÃO DO PROBLEMA, apresenta uma descrição do problema que será resolvido, bem como os fatores, parâmetros e ambientes associados, além de um levantamento de como o referido problema vem sendo resolvido pelos profissionais da área envolvida.

No terceiro capítulo, BASES METODOLÓGICAS PARA RESOLUÇÃO DO PROBLEMA, procura-se tratar da aplicação dos conteúdos vistos nas disciplinas estudadas, visando a resolução do problema proposto na introdução e fundamentado no Capítulo 2.

No quarto capítulo, MODELO PROPOSTO, é apresentada uma explicação a respeito dos *hardware*s e *software*s desenvolvidos, suas principais funcionalidades e as ferramentas utilizadas para sua elaboração.

No quinto capítulo, SIMULAÇÕES E RESULTADOS são mostrados as simulações e os resultados obtidos durante a execução do projeto, bem como, as dificuldades encontradas.

No sexto capítulo, CONSIDERAÇÕES FINAIS, é abordada a conclusão do trabalho e as sugestões para trabalhos futuros.

CAPÍTULO 2- APRESENTAÇÃO DO PROBLEMA

Segundo Walter (2003), restaurantes são ambientes onde o público vai para se refazer, tanto nutritivo quanto psicologicamente. Os restaurantes são conhecidos por servir alimentos e bebidas para seus consumidores, por meio de pagamento por este serviço. São vários os tipos de restaurantes e eles podem ser classificados pelo tipo de cozinha, tipo de serviço, tamanho do estabelecimento, ou outros critérios. Segue alguns exemplos:

• Restaurante clássico ou internacional – tem padrão requintado e o cardápio é composto de pratos da cozinha internacional e alguns nacionais.

• Churrascaria – restaurante especializado em churrasco (cozimento em brasa, em que a superfície fica dourada e o interior cozido). Geralmente, as carnes de churrasco são servidas à mesa e os demais pratos em bufê.

• Restaurante típico – caracteriza, de forma bem marcada, uma região ou um país. O cardápio e a decoração são característicos do local representado.

• Café – casa onde são servidos, além do café, chás, bebidas alcoólicas e lanches.

• Pub – do inglês *public house* - tipo de bar da Inglaterra. Serve bebidas e alguns pratos.

• Restaurante de hotel ou pousada – normalmente, serve refeições para os hóspedes. Alguns deles são abertos para o público externo. O serviço mais comum nesse tipo de restaurante é o de café-da-manhã.

• Refeitório de empresa – é um restaurante localizado dentro de uma empresa e serve refeições para os funcionários. Sua característica principal é, normalmente, o serviço simples e rápido.

Assumindo diferentes formas, a automação de sistemas pode estar presente em qualquer um desses tipos de restaurantes. Entretanto, a solução apresentada neste projeto é mais indicada para restaurantes com número grande de mesas e frequentadores, já que estes que normalmente teriam dificuldades para conseguir a atenção do garçom.

Segundo Castelli (2001), todas as pessoas que executam tarefas e as passam adiante são denominadas fornecedores, e as que recebem são chamadas clientes. Ou seja, cliente “é a próxima etapa do processo” e pode ser tanto o interno quanto o externo.

O cliente é o motivo pelo qual qualquer empresa prestadora de serviços existe, uma vez que sem ele não há prestação de serviços, e consequentemente não existirá a empresa prestadora de serviços. O cliente é o ponto de partida para uma empresa que deseja prosperar.

Para Giglio (1996 p: 14), “a satisfação do cliente é o princípio e o fim de nosso trabalho”. Procurar satisfazer o cliente implica em descobrir seus desejos, necessidades, exigências e satisfações de forma a manter o cliente permanentemente. Mais do que oferecer produtos de qualidade, é necessário um atendimento de qualidade.

“O cliente se conquista e se mantém com base na qualidade de atendimento”. (COBRA, 1993 p: 1). O proprietário do restaurante Vetan Chicago, Dought Roth, acertou ao dizer: “Um bom atendimento poderá salvar uma refeição ruim, entretanto uma refeição excelente jamais salvará um mau atendimento”. Maricato (2003) corrobora ao afirmar:

> O objetivo maior de uma empresa, é satisfazer o cliente, fazê-lo sair sorrindo e pensando em voltar, em recomendar o estabelecimento aos conhecidos, contente com a relação custo-benefício encontrada. Benefício, no caso de bares e restaurantes, inclui a qualidade dos produtos, a eficiência e a gentileza dos serviços, ambientação adequada, decoração, altura do som, iluminação, clima, limpeza, preço compatível,
> localização etc...(FONTE: Disponível em <http://pt.scribd.com/doc/49905729/12/Cliente>)

De acordo com o mesmo Autor, a excelência no tratamento do cliente enquanto norma de conduta é o princípio maior de uma empresa, e merece considerações específicas. Antes de tudo para servir otimamente o cliente é preciso antes conhecê-lo.

Segundo Zemke (1991), as características que diferem o atendimento de alta qualidade do cliente são: atenção pessoal, seguida por confiabilidade, presteza e competência dos funcionários. As empresas que prestam um bom atendimento ao cliente crescem 6% ao ano, já as que não prestam bom atendimento perdem 2% por ano de participação. Com isso podemos dizer que as empresas dispostas a prestar um atendimento diferenciado ao cliente lucram com isso. Aqueles sem vontade ou incapazes de atingir esse padrão não prosperam e não irão prosperar, e possivelmente nem irão durar muito tempo no mercado.

A era digital muda tudo, inclusive a relação do consumidor com as empresas. Não basta só acompanhar a evolução das tecnologias, para estar na onda de inovações que corre sobre os negócios é preciso conhecer o consumidor do futuro e dessa forma, apresentar-se como a empresa do futuro.

Após a participação no National Retail Federation 2011 – o maior evento de varejo do mundo, realizado em janeiro deste ano - especialistas brasileiros afirmam que a grande tendência dos empreendimentos é o foco no novo consumidor. Graças às novas tecnologias, o novo consumidor interage mais e tem mais informações.

Além disso, uma pesquisa recente do Institute for Business Value da IBM[1] mostra que a principal característica do novo consumidor é a inteligência. O estudo contou com a participação de 30.624 consumidores de 13 países diferentes. Entre informações mais específicas sobre região, geração e segmentos do cliente, o estudo aponta algumas tendências gerais do cliente do futuro:

- com as novas tecnologias de informação e comunicação, o consumidor está cada vez mais conectado com a empresa;

- os clientes estão mais exigentes do que nunca. O que eles mais querem é uma experiência personalizada de consumo. Não querem só comprar, querem ser servidos. E querem que o vendedor os conheça bem. O empreendedor deve estar preparado para customizar sua oferta de acordo com o perfil do cliente;

- os consumidores estão interconectados, trocando informações com milhões de pessoas em redes sociais e positivamente, isto possibilita o consumidor ajudar o empresário. Não bastasse a propaganda informal que um cliente satisfeito pode fazer, a pesquisa da IBM mostra que muitas pessoas estariam dispostas a colaborar com as empresas testando ou divulgando produtos e serviços que atendam suas necessidades.

Por fim a pesquisa da IBM[2] mostra que "Consumidor inteligente" é a definição do cliente do futuro. Ele sabe o que quer. Sabe obter informações sobre o que quer e sabe escolher os jeitos de comprar o que quer. Onipresente, exigente, poderoso e inteligente, ele quer, enfim, o bê-á-bá do consumo: preços bons, promoções, serviços eficientes, variedade e qualidade.

Contrastando com esta realidade, na maior parte dos restaurantes, bares e hotéis para chamar o garçom até a mesa, o cliente ainda deve ficar com o dedo levantado ou assobiar para o garçom até que este perceba que seu serviço está sendo solicitado. Esta realidade foge das tendências gerais do cliente do futuro, pois a interação do cliente com o estabelecimento é pequena, o atendimento é subjetivo, uma vez que depende da percepção do garçom em distinguir qual mesa fez a solicitação primeiro, além do que o atendimento pode ser demorado ou até mesmo o cliente pode passar pelo constrangimento de não ser visto.

Por conseguinte, diante das características do consumidor inteligente surge a necessidade de uma maneira mais planejada para solicitar o atendimento do garçom. Nesse

[1] Disponível em: http://www.ibm.com/expressadvantage/br/articles_general_industry_3Q7.phtml. Acessado em 25/10/2011
[2] Idem.

sentido, o presente trabalho propõe o desenvolvimento de um sistema que permite ao cliente chamar o garçom apertando um simples botão.

O projeto consiste na criação de um aparelho, que deverá ficar na mesa do cliente, com um botão para chamar o garçom. Assim que o cliente apertar este botão, o pino que está conectado ao microcontrolador será aterrado, o microcontrolador por sua vez fará o processamento de seu conjunto de instruções e por meio do transmissor irá encaminhar via ondas de rádio (RF) o número da mesa que está solicitando o pedido a um receptor conectado a outro microcontrolador PIC que através do conversor MAX232 fará com que o número desta mesa seja mostrado na tela de um computador para que todos os garçons possam visualizar qual mesa está solicitando o serviço. Assim que uma nova mesa solicitar o atendimento um sinal sonoro também será emitido.

Além disso, na tela do computador será simulada a organização das mesas do ambiente do estabelecimento. Nesta etapa será utilizada a linguagem de programação JAVA pois é considerada mais acessível para a programação de objetos gráficos.

Como existe um grande questionamento sobre os reais ganhos advindo dos investimentos em tecnologia da informação(TI)[3], antes de sugerir que esta solução possibilita melhorar o atendimento nas mesas, aumentar a produtividade dos garçons e reduzir o tempo de espera para a entrega dos pedidos é importante mostrar o resultado do estudo efetuado por Byrd & Marshall(1997) acerca do relacionamento entre investimentos da TI e desempenho da empresa.

O estudo baseou-se em dados de 350 empresas por um período de quatro anos, objetivando com isso abranger um período de tempo no qual se pudesse notar os efeitos das aplicações de TI, cujo retorno de investimentos muitas vezes têm um tempo de maturação maior do que um ano. Para a realização deste estudo, foi realizada análise do relacionamento entre variáveis de investimentos em TI e indicadores tradicionais de desempenho de empresas.

Algumas das principais conclusões e questionamentos resultantes deste estudo:

- O dispêndio na totalidade do pessoal de *staff* [4] mostrou relacionamento negativo com os resultados da empresa, pois se prioriza quadros de funcionários numerosos em detrimento de pessoal melhor qualificado.

[3] Tecnologia da Informação (TI), incluem os sistemas de informação, o uso de hardware e *software*, telecomunicações, automação, recursos multimídia, utilizados pelas organizações para fornecer dados, informações e conhecimento (LUFTMAN et al., 1993; WEIL, 1992).

[4] Staff: equipe de funcionários que dão apoio à execução das atividades

- Há necessidade de abordagem híbrida: qualitativa e quantitativa. Os estudos qualitativos focam casos de sucessos, mas muito seria aprendido estudando-se os casos de fracasso de implantação de aplicações de TI. Por outro lado, seria interessante estudar quanto que os estudos quantitativos não captam "o contexto organizacional", ou seja, não permitem compreender o contexto que se dá a aplicação.
- A importância de taxonomias e tipologias da estratégia, negócio, estrutura, cultura da empresa e da TI.

A principal conclusão que os autores chegaram após discutirem os aspectos acima, é que a pergunta *"aumento de investimentos em TI levam a um maior desempenho organizacional*?" não é adequada, pois o desempenho das pessoas que estarão trabalhando com essas tecnologias alteram fortemente essa proporção e afetam o desempenho organizacional. Por isso, para avaliarmos se o sistema proposto para os restaurantes é realmente eficaz, há necessidade de uma análise mais abrangente para chegarmos a uma conclusão.

Portanto, neste momento não é possível afirmar que irá ocorrer o aumento da eficácia do estabelecimento, entretanto, fundamentado em experiências anteriores e observação em restaurantes que utilizam algum tipo de automatização para o atendimento a clientes, é possível elencar algumas vantagens para o estabelecimento. Vantagens como o monitoramento constante dos pedidos, que poderão ser acompanhados pelo gerente da casa através do monitor; o direcionamento do atendimento do garçom, já que ele identificará de imediato quem está chamando e não precisará ficar perguntando ao cliente se precisa de algo mais e o bom atendimento até em ambientes separados, uma vez que é possível acompanhar os pedidos sem precisar deslocar as equipes.

Com um monitoramento constante, também é possível aumentar o faturamento da casa, em especial das bebidas, uma vez que o serviço contribui para a agilidade no atendimento.

Realizando o levantamento de como o referido problema vem sendo resolvido pelos profissionais da área envolvida encontramos em páginas da internet, várias soluções parecidas no mercado de trabalho, assumindo diferentes nomes desde "*Smart Bell*","Psiu Garçom", "Aceno Digital", "Garçom inteligente", entre outros .A automação está presente em diferentes etapas do processo do atendimento a mesas em restaurantes.

O "*Smart Bell*"[5] consiste em um dispositivo sem fio, que é colocado em cada mesa, um display que é fixado à parede e um display/Relógio para uso do garçom. Os dispositivos da mesa enviam um sinal digital para o display/Relógio que o interpreta e exibe o número da mesa que está chamando.

Cada display exibe simultaneamente até 3 números de mesa, e cada relógio exibe até 5 números simultaneamente. A figura 2.1 a seguir representa este dispositivo .

Figura 2.1- *Smart Bell*
(Fonte: http://www.5solucoes.com.br/)

Já o "Psiu Garçom" é composto por transmissores individuais sem fio posicionados nas mesas do estabelecimento e um display eletrônico, instalado em um local de fácil visualização. Para chamar o atendente, o cliente aciona um botão do transmissor localizado em sua mesa e seu número é exibido no painel, ao mesmo tempo em que emite um alarme sonoro avisando o funcionário sobre a solicitação.

O display do Psiu Garçom fica no balcão central do estabelecimento, facilitando a orientação dos garçons quanto à mesa que realizou o chamado. Entre outras funcionalidades, este sistema conta com o registro de todos os atendimentos feitos em uma central de dados que quantifica o tempo de espera para cada chamado por meio de gráficos gerados por um programa do próprio Psiu Garçom, repassados para um computador via conexão de cabo USB.

[5] Smart Bell- disponível em http://www.5solucoes.com.br/ Acessado em 05/10/2011

Figura 2.2-Psiu Garçom

(Fonte: Disponível em:< http://oexpressobandeirante.com/?p=8117> Acessado em 10/11/2011)

Também podemos citar a solução encontrada para o caso dos restaurantes a la carte. Tradicionalmente para restaurantes que trabalham dessa maneira, o garçom precisaria anotar os pedidos num pedaço de papel, entregá-los ao pessoal da copa e cozinha, servir e depois fechar a conta. Nesse processo, o profissional perde tempo que poderia ser usado para atender mais mesas. Além disso, os pedidos anotados podem vir com grafia ilegível ou incorreta ou mesmo extraviar-se e provocar falhas na entrega.

A automação encontrada para esta situação baseia-se no registro do pedido eletronicamente, seja com um micro-terminal (teclado e display) ou um coletor de dados(ver Figura 2.3). A informação é encaminhada por rádio frequência ou rede para a copa e/ou cozinha, onde há um terminal do tipo PC que imprime o pedido. Outro funcionário cuida da entrega, o que libera o garçom para atender um número maior de mesas. Esses mesmos dados são repassados para o caixa, onde são impressos durante o fechamento da conta.

Figura 2.3-Coletor de Dados

(Fonte: Disponível em:< http://oexpressobandeirante.com/?p=8117> Acessado em 10/11/2011)

Os mesmos coletores de dados ou micro-terminais usados pelos garçons em restaurantes também podem servir para as camareiras em hotéis e motéis registrarem e encaminharem os dados do uso de serviços como frigobar, sabonetes, toalhas, etc. De mesmo modo, as informações são transmitidas por radiofreqüência, inclusive para o fechamento da conta.

Essas automações melhoraram a qualidade do atendimento ao cliente e por isso, merecem um olhar atento do brasileiro, tendo em vista que o Brasil é a sexta economia do mundo, superando a Itália e vem atraindo continuamente investidores para o mercado interno que está altamente aquecido. Além disso, após nossa nomeação oficial para dois eventos mundiais, Copa do Mundo de 2014 e Olimpíadas de 2016, nos colocamos em posição ainda maior de evidência mundial. Hoje, mais do que nunca, somos obrigados a nos adequar rapidamente ao mundo Globalizado. A utilização do sistema deste projeto para restaurantes pode trazer maior conforto para o cliente e agilidade no atendimento, tornando a estadia dos consumidores no local mais confortável e por conseguinte tornando maior a chance de o estabelecimento agradar o consumidor.

CAPÍTULO 3- BASES METODOLÓGICAS PARA RESOLUÇÃO DO PROBLEMA

Este capítulo apresenta o referencial teórico e metodológico, úteis na solução apresentada. São expostos conceitos sobre a tecnologia e principais componentes que farão parte da solução. Dentre esses componentes podemos mencionar: microcontroladores, o padrão RS-232, o meio de transmissão e as linguagens de programação.

3.1- Microcontrolador

O microcontrolador (MCU ou μC) segundo Souza(2008) é um pequeno componente eletrônico munido de inteligência programável, o que permite que suas saídas possam ser controladas, fazendo com isso que o microcontrolador possa ser utilizado no controle de diversos periféricos como LEDs, *buzzers*, botões, sensores (luminosidade, temperatura, umidade, pressão, etc), d*isplays*, entre outros.

Para Souza(2008), a utilização do adjetivo "pequeno" se deve ao fato de o microcontrolador possuir em uma única pastilha de silício encapsulada(popularmente chamada de CI ou CHIP), todos os componentes necessários ao controle de um processo. Em outras palavras, o microcontrolador possui internamente memória de programa, memória de dados, portas de entrada e/ou saída paralela, timers, contadores, comunicação serial, PWM, conversores analógico-digitais, etc.

O que diferencia um microcontrolador de um microprocessador (MPU ou μP) é a quantidade de memória interna (programa e dados), a velocidade de processamento, a quantidade de pinos de entrada/saída (I/O), alimentação, periféricos, arquitetura e set de instruções.

Além disso, enquanto o microprocessador precisa de componentes como memória, *timers* e interfaces de entrada e saída de dados para o seu funcionamento, o microcontrolador como já mencionado, possui embutido todos os recursos necessários para a sua utilização. Devido a estes fatores, o microcontrolador é chamado de sistema computacional em um único circuito integrado (*on-chip computer*).

Outra diferença existente entre os dois é o fato de os microprocessadores serem utilizados em aplicações que exigem cálculos matemáticos complexos e com muita velocidade e, os microcontroladores serem utilizados onde a velocidade de processamento não é tão alta, por exemplo, em máquinas de lavar roupa, esteira rolante, etc.

Os microcontroladores são utilizados principalmente nas áreas automobilística, automação, robótica, entretenimento, médica, controle de tráfego e segurança. Os microcontroladores trabalham utilizando uma freqüência de *clock* muito baixa se comparados aos microprocessadores atuais, todavia, eles são apropriados para uma vasta área de aplicações triviais. Nessas aplicações o uso dos microprocessadores seria desnecessário, pois o preço seria maior e uma grande capacidade de processamento ficaria inutilizada.

O consumo de energia do microcontrolador, em geral, é relativamente baixo, normalmente na casa dos *miliwatts* e muitos modelos possuem a habilidade de entrar em modo de espera (*Sleep*) aguardando por uma interrupção ou evento externo, como por exemplo, o acionamento de um botão, ou um sinal que chega por uma interface de dados. O consumo destes microcontroladores em modo de espera pode chegar na casa dos *nanowatts*, tornando-os ideais para aplicações onde a exigência de baixo consumo de energia e um fator decisivo para o sucesso do projeto. (SOUZA, 2003)

Os microcontroladores PIC são fabricados pela Microchip Technology, e processam dados de 8, 16 e 32 bits, tendo uma grande variedade de modelos, com recursos de programação por memória flash, EEPROM e OTP. Neste projeto é utilizado o microcontrolador PIC16F628A que tem as seguintes características:

- 18 pinos;

-possui somente 35 instruções no seu microcódigo;

- sinal de clock com freqüência até 20 MHz;

-memória de programa do tipo Flash de 2048 words (1 word= 32 bits);

-224 bytes de memória RAM para dados;

-128 bytes de memória EEPROM para dados;

-instruções de 14 bits com 200 ns de tempo de execução;

- dados de 8 bits por endereço de memória;

- 15 registradores especiais;

- 16 pinos os quais podem ser configurados como entrada e/ou saída;

- outras características especiais como programação *in-circuit* serial, proteção por código , watchdog timer(temporizador cão de guarda), módulo CCP, comparador interno, USART.

A pinagem do PIC16F628A pode ser conferida na figura 3.1. Os pinos RB1 e RB2 são utilizados para a comunicação serial (USART) e os pinos V_{SS} e V_{DD} compõem a parte destinada a alimentação e são fundamentais para o funcionamento desse projeto.

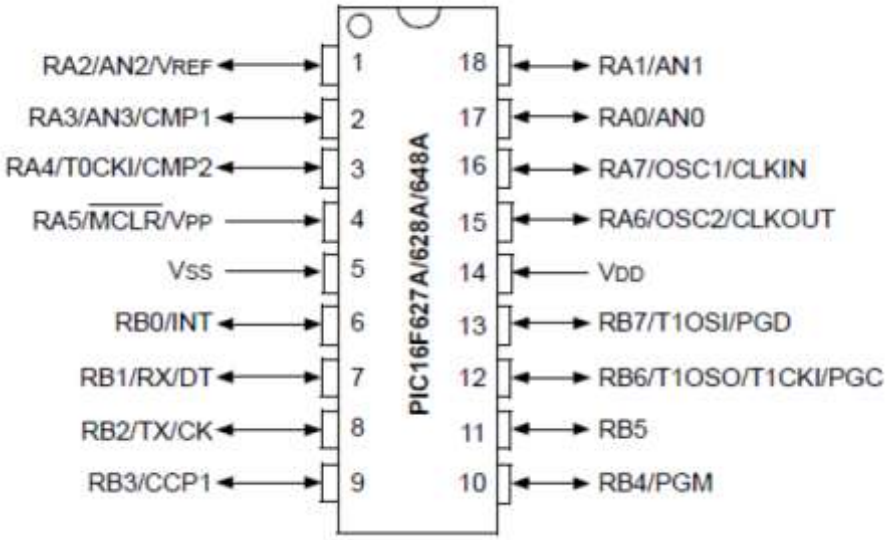

Figura 3.1 –Pinagem do PIC16F628A

(FONTE: Microchip, 2007)

O Quadro 3.1, mostra as funções e descrições de cada um desses pinos.

Pino	Função	Tipo	Descrição
1	RA2/AN2/Vref	Entrada/saída	PORTA bit 2 / Entrada do comparador analógico / Saída da tensão de referência
2	RA3/AN3/CMP1	Entrada/saída	PORTA bit 3 / Entrada do comparador analógico / Saída comparador 1
3	RA4/T0CKI/CMP2	Entrada/saída	PORTA bit 4 / Entrada de clock externo do timer 0 / Saída comparador 2. *Esse pino possui saída com **dreno aberto***
4	RA5/MCLR/VPP	Entrada	PORTA bit 5 / Reset CPU / Tensão de programação
5	VSS	Alimentação	Terra
6	RB0/INT	Entrada/saída	PORTB bit 0/ Entrada de interrupção externa
7	RB1/RX/DT	Entrada/saída	PORTB bit 1/ Recepção USART (modo assíncrono) / Dados (modo síncrono)
8	RB2/TX/CK	Entrada/saída	PORTB bit 2/ Transmissão USART (modo assíncrono) / Clock (modo síncrono)
9	RB3/CCP1	Entrada/saída	PORTB bit 3 / Entrada ou saída do módulo CCP
10	RB4/PGM	Entrada/saída	PORTB bit 4 / Entrada de programação **LVP***
11	RB5	Entrada/saída	PORTB bit 5
12	RB6/T1OSO/T1CKI/ PGC	Entrada/saída	PORTB bit 6 / Entrada do oscilador do TMR1 / Entrada de clock do TMR1 / Clock na programação **ICSP***
13	RB7/T1OSI/ PGD	Entrada/saída	PORTB bit 7 / Entrada do oscilador do TMR1 / Dados na programação ICSP
14	VDD	Alimentação	Alimentação positiva (3V a 5V)
15	RA6/OSC2/CLKOUT	Entrada/saída	PORTA bit 6 / Entrada para cristal oscilador / Saída de clock
16	RA7/OSC1/CLKIN	Entrada/saída	PORTA bit 7 / Entrada para cristal oscilador / Entrada de clock externo
17	RA0/AN0	Entrada/saída	PORTA bit 0/ Entrada do comparador analógico
18	RA1/AN1	Entrada/saída	PORTA bit 1/ Entrada do comparador analógico
*LVP - Baixa voltagem de programação. *ICSP - Programação in-circuit			*Dreno aberto - Uma fonte de alimentação externa deve ser aplicada ao pino.

Quadro 3.1- Funções e descrições dos pinos do PIC16F628A.
(FONTE: Zanco, 2005, p.37)

O uso deste microcontrolador torna possível a realização de diversas tarefas necessárias ao funcionamento do sistema. Entre essas tarefas, pode-se destacar a transmissão e recepção de dados via comunicação RF, entre outros. No capitulo 4 , são fornecidos mais detalhes sobre o modo de funcionamento dessas tarefas no contexto do projeto proposto.

No diagrama de blocos (ver Figura 3.2) podem ser visualizadas as diferentes partes que compõem o microcontrolador PIC16F628A. Podemos observar a ULA diretamente ligada ao registrador W. No canto superior esquerdo está a memória de programa, e logo após, saindo deste bloco está o barramento de 14 bits. Ainda na parte superior, está localizado o contador de linha de programa e a pilha de 8 níveis Ao centro da imagem está a memória de dados que possui um barramento de 8 bits e o registrador de status que armazena algumas informações importantes sobre as operações aritméticas da ULA. Do lado direito podemos ver as portas e todos os seus pinos de I/O. Já na parte inferior, localizam-se o comparador, a EEPROM, os timers, o módulo CCP e a porta serial.

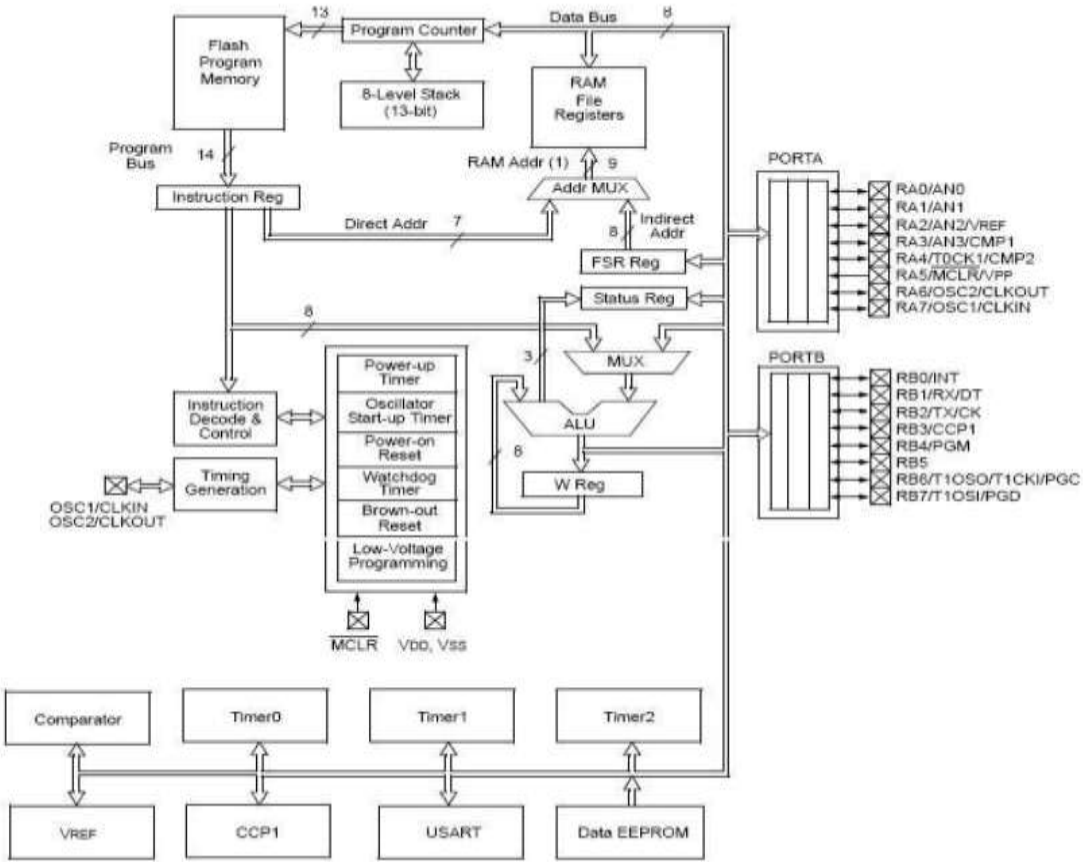

Figura 3.2 –Pinagem do PIC16F628A

(FONTE: Microchip, 2007)

3.2- Linguagens de Programação

Para podermos escrever (gravar) em um microcontrolador é necessário um programa, um compilador, um *software* gravador e uma gravadora.

O microcontrolador possui o controle de todas as ações ocorrentes no sistema, mas para isso, é necessário desenvolver uma programação capaz de identificar e tratar essas informações. Para tal finalidade, nesse projeto foi escolhida a Linguagem C, devido à facilidade no uso dos microcontroladores PIC. A utilização dessa linguagem, permite ao programador se preocupar apenas com a programação da aplicação em si, já que para integração dos dispositivos, a linguagem C oferece uma grande biblioteca de apoio tanto em livro como na própria internet. Além do mais, essa linguagem permite certa complexidade para o uso de funções muito maior que o Assembly, por exemplo. Acrescentando-se aos fatores expostos acima, C é uma linguagem que qualquer Engenheiro de Computação deve saber e ter noção de seus conceitos, pois esta linguagem serviu como base para a criação de inúmeras outras. Seu conhecimento facilita bastante o entendimento de novas linguagens mais avançadas (JAMSA, 1999).

No desenvolvimento do projeto também foi utilizada a linguagem de programação JAVA para a criação da aplicação desktop com a reprodução do ambiente do estabelecimento na tela do computador. Java foi escolhido devido entre outras características, ser uma linguagem robusta e fácil de ser implementada, e ser orientada a objetos. Segundo Deitel (2006), um objeto pode ser entendido assim como na vida real, por exemplo um carro, um telefone ou uma casa. Cada objeto possui uma classe que contém atributos e métodos que o caracterizam.

3.3– IDE CCS C Compiler

O IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado) CCS C Compiler, fornecido pela CCS Inc., é um ambiente de desenvolvimento da linguagem de programação C voltado especialmente para microcontralores PIC. Ele integra os operadores padrões da linguagem C e funções específicas para os registradores PIC, proporcionando aos desenvolvedores uma poderosa ferramenta para a programação do *hardware*.

Por meio deste ambiente, é possível o desenvolvimento e compilação do firmware dos microcontroladores PIC e neste projeto ele foi utilizado com este objetivo. Além disso, nele também é possível depurar o código desenvolvido.

Para Aho (2008) “um compilador é um programa que recebe como entrada um programa em uma linguagem de programação – a linguagem *fonte* – e o traduz para uma programa equivalente em outra linguagem – a linguagem *objeto*; ver . Uma das funções principais do compilador é relatar diferentes tipos de erros no programa fonte detectados durante esse processo de tradução.”

Figura 3.3 - Um compilador

(FONTE: Aho, 2008, p. 1)

O programa desenvolvido na linguagem C por meio do CCS C Compiler, é traduzido para a linguagem de máquina que os microcontroladores PIC podem interpretar.

3.4- Kit de Gravação PICkit2

A gravadora é o dispositivo responsável por inserir os comandos de programação em um microcontrolador, pois os microcontroladores não possuem interface direta com o computador. Deste modo, a gravadora pode ser definida como “um circuito que faz a interface entre o microcomputador e o microcontrolador, realizando as tarefas de programação e configuração” (ATMEL, 2008).

O princípio de gravação dos dados é o mesmo para todas as gravadoras, o que difere é o *hardware* escolhido pelo desenvolvedor. Algumas gravadoras possuem a fonte na própria placa e em outras deve ser utilizada a tensão existente na porta de comunicação do micro (em geral, aproximadamente 15mA).

O processo de inserção dos dados pode se dar através de portas seriais ou paralelas, pois por intermédio dessa interface transfere-se o código programado para a memória flash do

microcontrolador. As gravadoras são estruturadas a partir do microcontrolador escolhido e podem ser compradas ou construídas. Nesse projeto optou-se pela compra da gravadora compatível com o microcontrolador. O equipamento utilizado no desenvolvimento desse projeto foi o PICkit2 composto por um circuito com conectores USB e ICSP, um soquete ZIF de 40 pinos e um cabo USB para conexão com o computador conforme ilustra a Figura 3.4.

Segundo RobóticaSimples (2009), o gravador de PIC/EEPROM USB, integra-se ao ambiente de desenvolvimento integrado (IDE) MPLAB, grava Firmwares criados por qualquer linguagem de programação, identifica o microcontrolador a ser gravado, possui conector ICSP (In-Circuit Serial ProgrammingTM) que permite gravar os dados e comandos do seu PIC diretamente em sua placa, conta ainda com Analisador Lógico de 3 canais e conversor RS232<=>TTL sobre USB que permite comunicar o seu computador diretamente com PIC.

Figura 3.4 - Gravador PICkit2

(FONTE: Braga ,2010, p. 68)

O *software* PICkit2 Programmer(ver Figura 3.5) é desenvolvido pela Microchip Inc. para plataformas Windows, Linux e MAC OS, nele é possível o controle das funções do gravador tais como: ler, gravar, apagar e verificar o microcontrolador.

O programa também identifica o microcontrolador conectado, fornecendo ao usuário funções específicas que cada um requer.

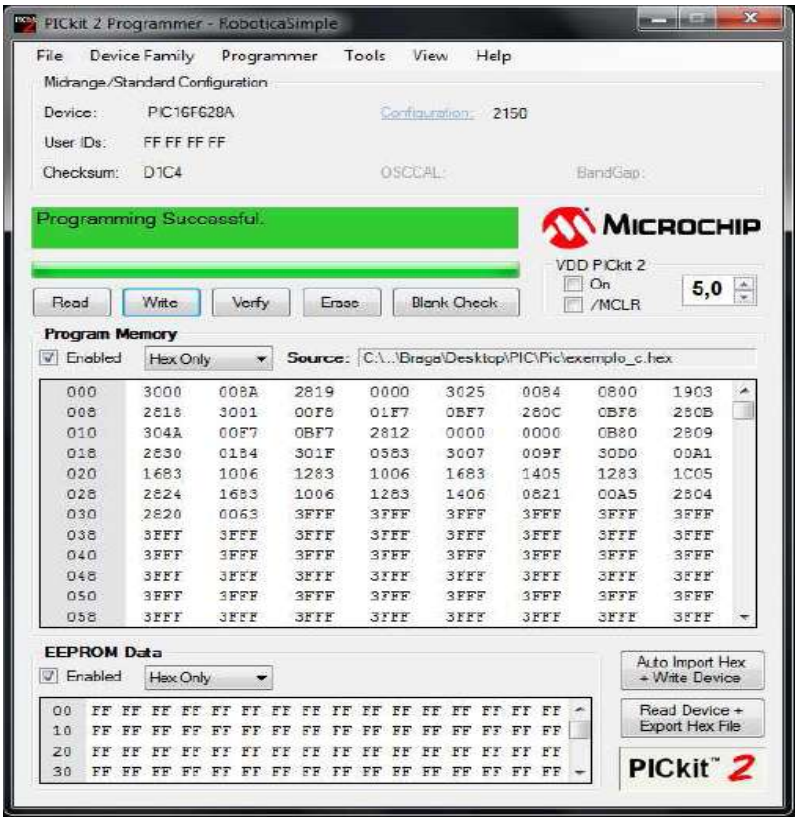

Figura 3.5 - PICKit2 Programmer

(FONTE: Braga, 2010, p. 69)

3.5- Transmissão dos Dados

Em determinadas situações faz-se necessária a transmissão de dados entre dois dispositivos em um meio que não utilize meio físico (fios), para isso podemos utilizar várias técnicas, como o infravermelho ou ainda ondas de rádio. A comunicação via ondas de rádio, portanto, consiste em conectar dois ou mais equipamentos fazendo-se o uso da radiofrequência. Por meio de técnicas apropriadas é possível realizar qualquer tipo de comunicação utilizando as ondas de rádio.

Para o estabelecimento da conexão via rádio podem ser utilizados alguns módulos prontos na forma de "módulos híbridos". Para o correto funcionamento da transmissão é necessário um módulo de transmissão (TX) e ou de recepção (RX). Estes módulos podem ser adquiridos com frequências de trabalho pré-definidas como: 315 Mhz, 418 Mhz e 433 Mhz (outros módulos podem trabalhar com frequências diferentes). A sua modulação é AM (modulação de amplitude).

Devido às técnicas de transmissão via radiofrequência foi possível o surgimento da radiodifusão, do telefone celular, do GPS, das comunicações a longas distâncias, por meio de

satélites e de sistemas em radiovisibilidade, e tantos outros serviços de telecomunicações (TANENBAUM, 1994).

Atualmente, existem diversos satélites orbitando pelo nosso planeta, com o objetivo de retransmitir informações ininterruptamente entre pontos distantes da terra, sem sofrer com interferências físicas. Por meio destes aparelhos foi possível o desenvolvimento de novos padrões, chamados protocolos de comunicação, sempre crescendo e inovando, pois, são utilizados tanto para codificação de sinais de televisão (áudio e vídeo), como para as conexões com a internet.

Por serem fáceis de percorrerem longas distâncias e penetram em prédios com facilidade, as ondas de rádio frequência são largamente utilizadas para comunicação, seja em ambientes fechados ou abertos. As ondas de rádio também são onidirecionais, o que significa que elas correm todas as direções a partir da origem; portanto, o transmissor e o receptor não precisam estar cuidadosa e fisicamente alinhados (TANENBAUM, 1994).

Como já mencionado, para a implementação de um sistema de transmissão de dados por ondas de rádio frequência é preciso, principalmente, módulos de rádio frequência responsáveis pela transmissão e recepção dos dados via ondas de rádio.

Para isso, utiliza-se os módulos híbridos que são circuitos completos do transmissor e do receptor já montados com componentes SMD[6] numa placa muito pequena, que pode ser encaixada e soldada diretamente na placa principal. Como estes módulos são fabricados em série por um processo muito preciso que inclui o ajuste de frequência com uso de laser, tem-se a garantia de que o sinal de receptor pode ser recebido pelo mesmo sem a necessidade de ajustes.

Neste projeto o circuito completo faz com que o transmissor envie uma sequên-cia de pulsos elétricos. Cada sequência contém um pequeno grupo de pulsos de sincronização, seguido pela sequência de pulsos de dados. O segmento de sincroni-zação alerta o receptor para a informação recebida e o segmento de pulso informa à antena o que é a nova informação. O transmissor envia ondas de rádio que oscilam com uma frequência de 433.000.000 ciclos por segundo (433 MHz). Enquanto isso o receptor monitora constantemente a frequência designada (433 MHz) à procura de um sinal. Quando o receptor recebe as ondas de rádio do transmissor, ele envia o sinal para filtros internos, que bloqueiam quaisquer outros sinais captados pela antena que estejam fora da frequência de 433 MHz. O

6

Dispositivo montado em superfície – uma nova tecnologia que tem por objetivo reduzir o espaço ocupado pelos tradicionais componentes (resistências, diodos, transistores e CI's). Disponível em:< http://www.angelfire.com/on/eletron/smd.html>; Acessado em 7 de Setembro de 2011

sinal remanescente é convertido novamente em uma sequência de pulsos elétricos decodificados e enviados ao microcontrolador.

Além da comunicação por meio de ondas de radio frequência, também foi utilizado neste projeto a comunicação de dados por meio de um cabo serial que utiliza o padrão RS-232 para a comunicação do *hardware* do receptor com o microcomputador.

3.5.1-Padrão Serial RS-232

O RS-232 (Recommended Standard 232) é um padrão de sinal desenvolvido pela Electronic Industries Association – EIA e ITU V.24/V28, partes interessadas em especificar a interface serial entre equipamentos de terminal de dados (Data Terminal Equipment – DTE) e equipamentos de comunicação de dados (Data Communications Equipment – DCE). DCE são dispositivos como modem, plotter, etc., enquanto DTE são computadores ou terminais. Para Alcântara (2008), ainda que seja bem antigo, este padrão continua sendo bem utilizado por sua simplicidade e confiabilidade".

A transmissão serial é caracterizada pelo envio sequencial dos bits. Um a um, os bits são enviados de forma assíncrona, ou seja, o transmissor e receptor precisam efetuar o controle de tempo para saber o início e o fim de cada bit.

Na sua forma padrão o RS-232 utiliza dois sinais de controle, o RTS (ready to send) e o CTS (clear to send) para efetuar o controle de fluxo via *hardware*. Basicamente, quando o transmissor deseja começar um envio ele sinaliza através do pino RTS. O receptor, ao perceber que o transmissor deseja enviar algum dado, prepara-se para recebe-lo e seta o pino CTS. Apenas depois de receber o sinal CTS o transmissor pode começar a transmissão. Para cada byte transmitido existem bit de start e stop; o mais comum é utilizar-se 1 bit de início (start bit) e 1 bit de parada (stop bit), mas é possível encontrar aplicações que utilizam 1,5 ou 2 bit de início/parada. A Figura 3.6 ilustra a transmissão de um byte.

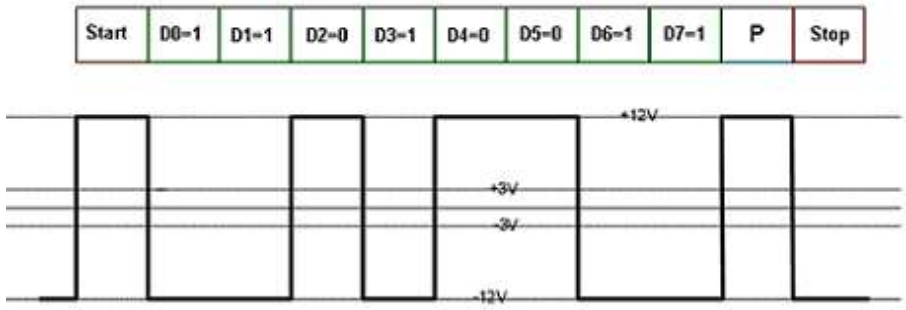

Figura 3.6 - Transmissão de um byte em RS-232

(FONTE: http://www2.eletronica.org/artigos/eletronica-digital/padrao-serial-rs-232)

Por apresentar uma transmissão assíncrona, no padrão RS 232 para que seja possível a identificação dos bits, é necessário que seja configurada a taxa de transmissão (baud rate) nos dispositivos.

O baud rate informa quantos bits no período de um segundo serão transferidos na linha. A unidade do baud rate é dada por bits por segundo (bps), os mais utilizados são 2400, 4800 e 9600 bps.

No transmissor o envio basicamente resume-se à enviar um bit de início, aguardar um tempo, e enviar os próximos 8 bit + bit de parada, com o mesmo intervalo de tempo entre eles. No receptor, após a primeira borda de descida (nível lógico de “1″ para “0″) (start bit) o receptor sabe que uma sequencia de mais 8 bit de dados + bit de parada chegará. Por conhecer a taxa de transmissão, tudo que o receptor precisa fazer é aguardar o tempo de transmissão entre cada bit e efetuar a leitura. Após receber o bit de parada, a recepção encerra-se e ele volta à aguardar o próximo start bit.

Nos microcontroladores existe um periférico denominado UART (*Universal Asynchronous Receiver Transmitter*), que é o responsável pelo controle da transmissão e das interrupções no momento de recebimento de um byte.

Segundo Alcântara (2008), “na interface RS232 o nível lógico "1" corresponde à uma tensão entre -3V e -12V e o nível lógico "0" à uma tensão entre 3V e 12V. Valores de tensão entre -3V e +3V são indefinidos e precisam ser evitados”. Todavia, o microcontrolador utilizado neste projeto não utiliza os níveis de tensão do padrão RS-232 e foi necessário um circuito adicional de conversão de níveis RS-232/TTL. O circuito integrado mais comum para efetuar essa conversão é o MAX232 (Figura 3.7).

Figura 3.7 - Circuito integrado conversor de níveis TTL/RS-232

(FONTE: Braga, 2010, p. 56)

3.5.2 – Cabo Serial DB9

A função principal deste cabo é para comunicações seriais RS-232 e no projeto foi utilizado para ligar o computador com o receptor.

O conector possui nove pinos, conforme ilustra a Figura 3.8 e as especificações de cada pino são listadas no Quadro 3.2.

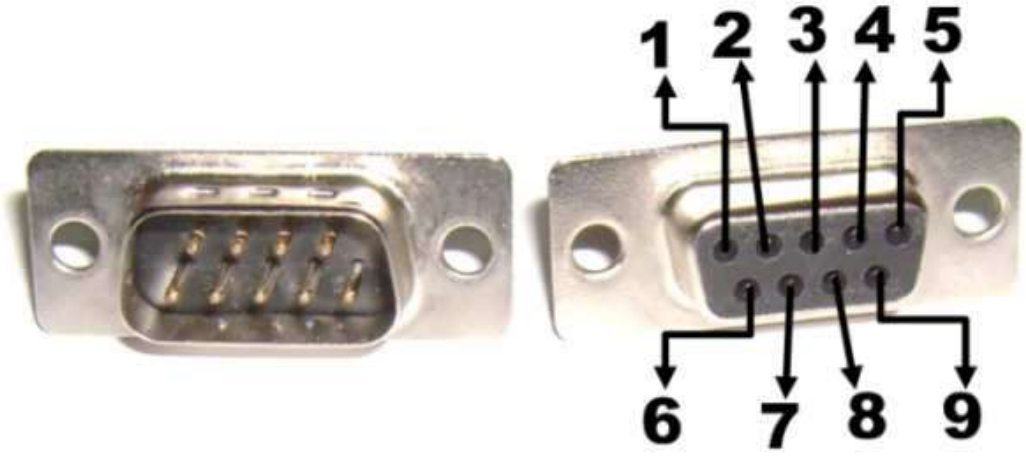

Figura 3.8 - Conector DB9
(FONTE: Braga, 2010, p. 53)

Número	Nome	Designação
1	CD – Carrier Detect	Detecção de portador
2	RXD – Receive Data	Recepção de dados
3	TXD – Transmit Data	Transmissão de dados
4	DTR – Data Terminal Ready	Terminal pronto
5	GND – Signal Ground	Massa lógica
6	DSR – Data Set Ready	Dados prontos
7	RTS – Request to send	Pedido de emissão
8	CTS – Clear to send	Empréstimo a emitir
9	RI – Ring Indicator	Indicador de campainha elétrica

Quadro 3.2 - Pinos DB9
(FONTE: Braga, 2010, p. 54)

CAPÍTULO 4- MODELO PROPOSTO

Definidos os conceitos, teorias e apresentadas as técnicas e dispositivos, é iniciado o desenvolvimento. A fase de desenvolvimento consiste, de modo geral em executar as tarefas previstas em planejamento e chegar ao produto final.

O ponto de partida de todo desenvolvimento de projeto é o atendimento aos objetivos propostos, para que se possa corresponder às expectativas. A constante observação dos requisitos deve ser um direcionamento que não pode ser esquecido. A escolha dos componentes leva em conta esse fator, portanto, os objetivos propostos no início do projeto devem ser revisados.

4.1 – Apresentação Geral Do Modelo Proposto

Na maior parte dos restaurantes, bares e hotéis, para chamar o garçom ou outro atendente até a mesa, o cliente deve ficar com o dedo levantado, fazer assobios, etc . Esta realidade foge das tendências gerais do cliente do futuro, uma vez que a relação do cliente com o estabelecimento é pequena, o atendimento é subjetivo, pois depende da percepção do garçom em distinguir qual mesa fez a solicitação primeiro, além do que o atendimento pode ser demorado ou até mesmo o cliente pode passar pelo constrangimento de não ser visto.

Partindo deste contexto, foi construído um protótipo utilizando microcontroladores da família PIC16F, que permite ao cliente chamar o garçom apertando um simples botão. Assim que o botão for apertado, o número da respectiva mesa irá aparecer na tela do monitor de um computador e um sinal sonoro será emitido. Foram implementados dois *hardware*s transmissores que ficarão nas mesas do cliente e um *hardware* receptor. A solução desenvolvida é ilustrada na Figura 4.1.

Assim que o botão for pressionado, o microcontrolador processa a informação e envia os dados referentes ao número da mesa para que o módulo transmissor RF realize a modulação. Este por sua vez transmite as informações ao módulo receptor RF que decodifica a informação e em seguida transmite ao microcontrolador. Antes de a informação chegar ao monitor, é necessário que ela passe por um conversor, que neste projeto é o MAX232. O conversor recebe o sinal do microcontrolador no padrão TTL e o converte no padrão RS-232 para que possa ser entendido pelo microcomputador.

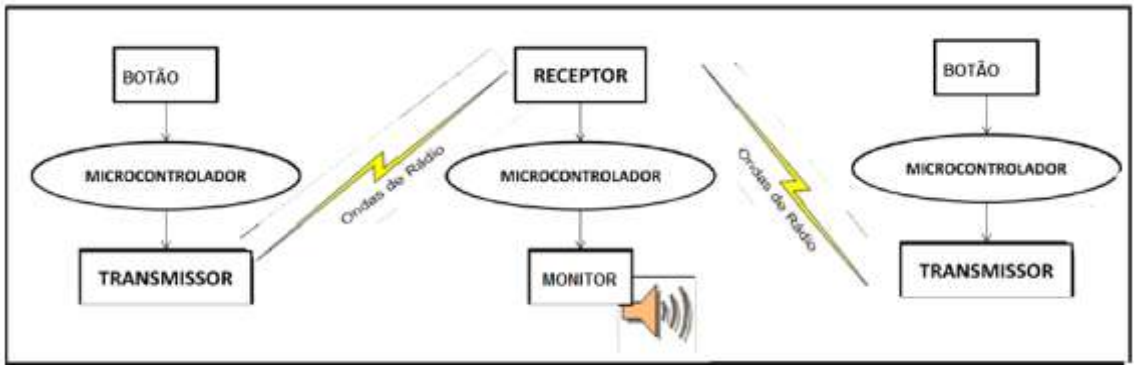

Figura 4.1- Visão Geral do Projeto

A seguir são mostradas as etapas para se chegar à solução geral, mostrando-se o funcionamento e uso de cada componente.

4.2 – *Hardware*

O *hardware* consiste de toda a parte física do projeto, e considera-se tanto as montagens mecânicas, como as ligações elétricas e os circuitos eletrônicos. O *hardware* corretamente especificado e integrado permite a programação das funcionalidades necessárias no projeto.

4.2.1 *Hardware* Módulo Receptor

O módulo receptor é o *hardware* que interliga os dispositivos das mesas dos clientes por meio de comunicação sem fio em radiofrequência com o computador através da comunicação serial RS-232.

Na montagem deste módulo, quatro componentes principais foram utilizados para realizar a tarefa descrita acima. São eles:

1. Receptor RF RXD1 433MHz
2. Microcontrolador PIC16F628A;
3. Circuito Integrado MAX232;
4. Monitor do Computador

O processo de recepção/transmissão é ilustrado na Figura 4.2.

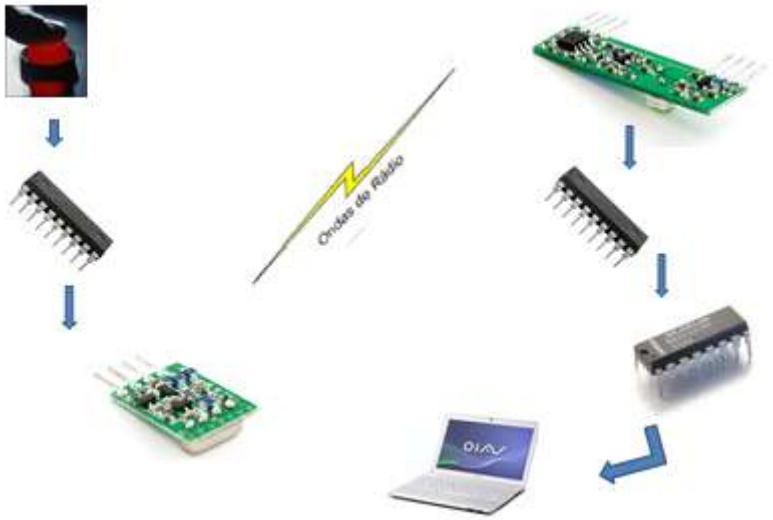

Figura 4.2- Processo de recepção/transmissão

4.2.1.1- Receptor RF RXD1 433MHz

É o componente do projeto responsável pela recepção dos dados enviados pelo dispositivo das mesas utilizando ondas de rádio operando na faixa de frequência de 433.92 MHz. O receptor tem como característica principal monitorar constantemente a frequência designada (433.92 MHz) à procura de um sinal no ambiente. Quando o receptor recebe as ondas de rádio do transmissor, envia o sinal para filtros, que bloqueiam quaisquer outros sinais captados pela antena que estejam fora da frequência de 433 MHz. O sinal remanescente é convertido novamente em uma sequência de pulsos elétricos decodificados e enviados ao microcontrolador.

Na Figura 4.3, são apresentadas as dimensões físicas do receptor em milimetros e a especificação técnica de seus pinos. Existem oito pinos, quatro em cada extremidade, dos quais, cinco são responsáveis pela alimentação do transmissor e o os outros três pela entrada e saída de dados recebidos.

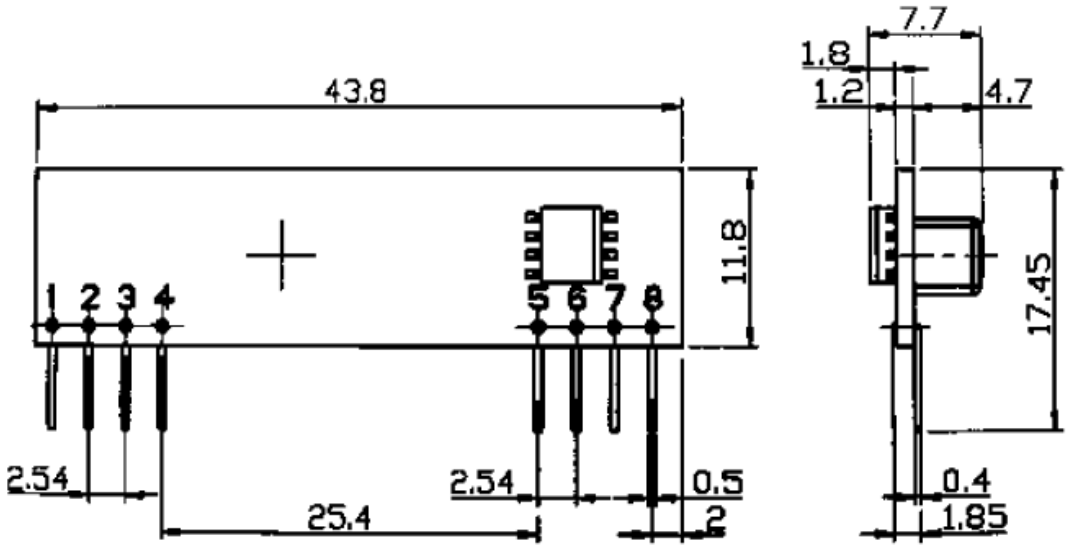

Figura 4.3 - Dimensões Físicas do Receptor. Fonte: KEYMARK, 2003

Os pinos do receptor foram conectados da seguinte maneira:

- Pinos 1, 6 e 7 – Terra;
- Pino 2 – Dados;
- Pinos 4 e 5 – 5V;
- Pino 8 – Antena.

Apesar dos vários pinos, a ligação do receptor é basicamente a alimentação, os dados e a antena.

O pino 2 é onde o receptor dispõe os dados recebidos por radiofrequência já demodulados e prontos para serem interpretados pelo microcontrolador. Ele é ligado ao pino RX do PIC (pino 7).

No pino 8 foi ligado uma antena para melhorar a recepção das ondas eletromagnéticas.

4.2.1.2 – PIC16F628A

O microcontrolador PIC foi ligado de forma a receber os dados do receptor e encaminhá-los ao MAX232. Nele também foram conectados um LED indicador de presença de corrente elétrica no circuito, além do resistor de 10KΩ e capacitor de 1pF recomendados pela Microchip Inc.

Os pinos do microcontrolador ficaram conectados da seguinte forma:

- Pino 4 – Resistor 10KΩ e 5V;

- Pino 5 – Terra;
- Pino 7– Pino 2 de RXD1;
- Pino 8 – Pino 11 do MAX232;
- Pino 9 – LED vermelho indicativo de transmissão e resistor
- Pino 14 – 5V e capacitor de 1pF;

O pino 4 do PIC é para *reset* do microcontrolador. O resistor conectado a ele é do tipo *pull-up*, que eleva a tensão do pino ao nível alto (5V) evitando que ocorram *resets* inesperados.

Os pino 5 e 14 são de alimentação do microcontrolador e são conectados a um capacitor para eliminar ruídos da fonte externa.

O pino de recepção de dados (RX) do microcontrolador é o pino 7 e os dados provenientes do RXD1 são recebidos por este pino. Já o pino 8 é o TX do microcontrolador e está conectado ao pino 11 do MAX232.

4.2.1.3- Comunicação Serial com MAX232

Devidos aos computadores atuais não possuírem mais portas seriais, para efetuar a conexão do computador com o *hardware* receptor, foi necessário um circuito conversor TTL/Serial e um cabo conversor de serial para USB, ver Figura 4.4, adquirido em lojas de informática. O circuito conversor MAX232 foi ligado a este cabo, utilizando um conector DB9 fêmea, conectado aos pinos 2,3 e 5 - RX, TX e terra respectivamente do CI MAX232.

Figura 4.4 - Conversor USB – Serial

(FONTE: www.mensis.com.br)

O circuito MAX232 foi montado de acordo com o *datasheet* do fabricante e ficou com a seguinte configuração:

- Pinos 1 e 3 – Capacitor de 1µF;
- Pinos 4 e 5 – Capacitor de 1µF;
- Pino 6 e GND – Capacitor de 1µF;
- Pinos 2 e 16 – Capacitor de 1µF;
- Pino 15 – Terra;
- Pino 16 – 5V;
- Pino 11 – Pino 8 do PIC16F628A;
- Pino 13 – Pino 3 do DB9;
- Pino 14 – Pino 2 do DB9;
- Pino 15 – Pino 5 do DB9;

A Figura 4.5 abaixo ilustra um esquema elétrico interessante para os testes com a comunicação serial RS-232 entre o microcontrolador PIC16F628A e um PC. O circuito integrado MAX232 existente permite adaptar os níveis de tensão da norma RS-232 para o padrão TTL, compatível com o microcontrolador PIC16F628A, e vice-versa.

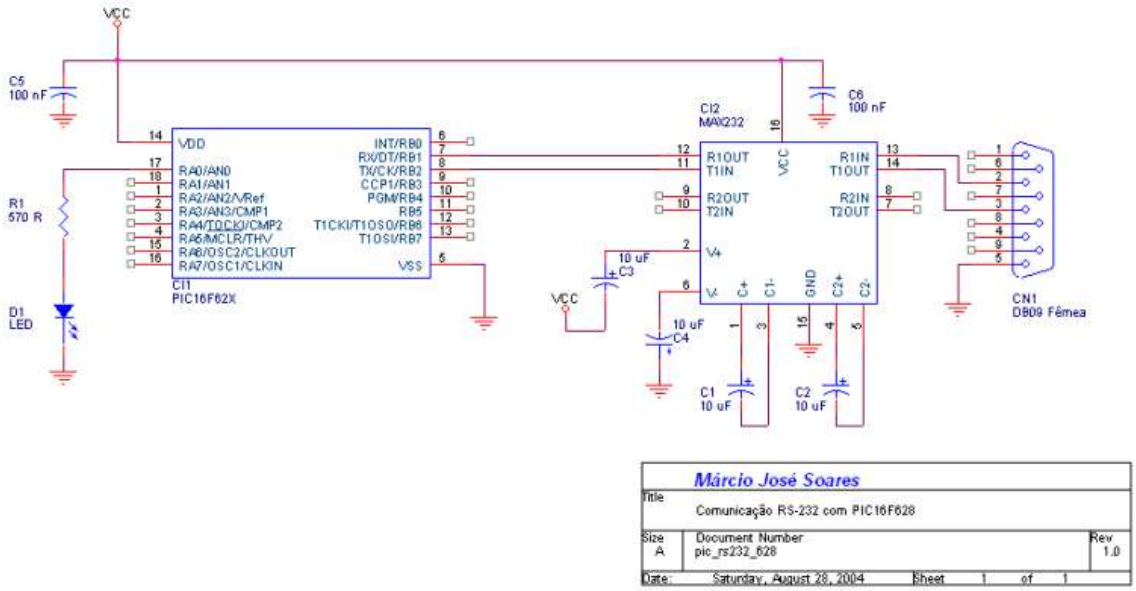

Figura 4.5 - Representação do circuito MAX232

(FONTE: Disponível em : http://www.arnerobotics.com.br/eletronica/comunicacao_rs232_PIC.htm Acessado em 20/10/2011)

Já a Figura 4.6 ilustra o circuito MAX232 montado na *protoboard*.

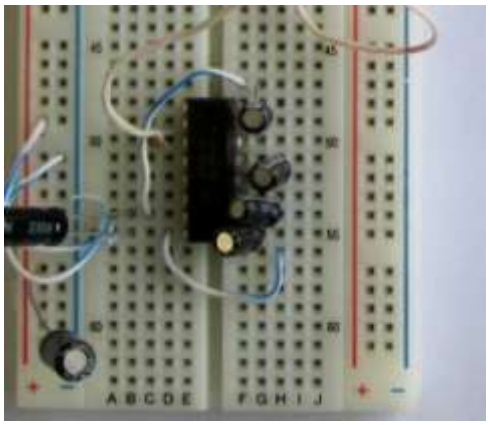

Figura 4.6 - Circuito MAX232 na *protoboard*

(FONTE: Autora)

4.2.1.4- Alimentação

Para alimentar o módulo receptor, uma fonte externa de 16.5VDC e 1A foi utilizada. Como a tensão da fonte era superior a tensão de trabalho dos demais componentes, era preciso um regulador de tensão que a ajustasse para 5VDC.

O regulador de tensão utilizado foi o LM7805 e foi ligado seguindo instruções recomendadas no *datasheet* do fabricante. O regulador possui 3 pinos, um de entrada, terra e uma saída. A entrada suporta tensões de 6VDC até 35VDC e faz a regulagem para 5VDC.

Como recomendação do fabricante para eliminar ruídos provenientes da rede elétrica, foi utilizado um capacitor de 100µF/50V entre o pino de entrada e terra do LM7805 e outro capacitor de 10µF/50V entre a saída e o terra. Neste circuito também foi ligado um LED para indicar quando o mesmo estiver ligado. O LED é ligado em série com um resistor de 220Ω entre os terminais 2 e 3 do regulador.A Figura 4.7 representa o esquemático no Proteus da ligação do regulador.

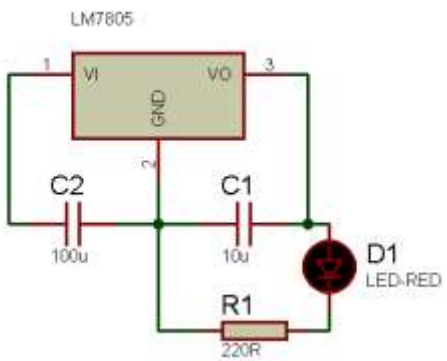

Figura 4.7 - Ligação do regulador de tensão (FONTE: Autora)

Na figura 4.8 é ilustrado os dispositivos utilizados para o circuito de alimentação.

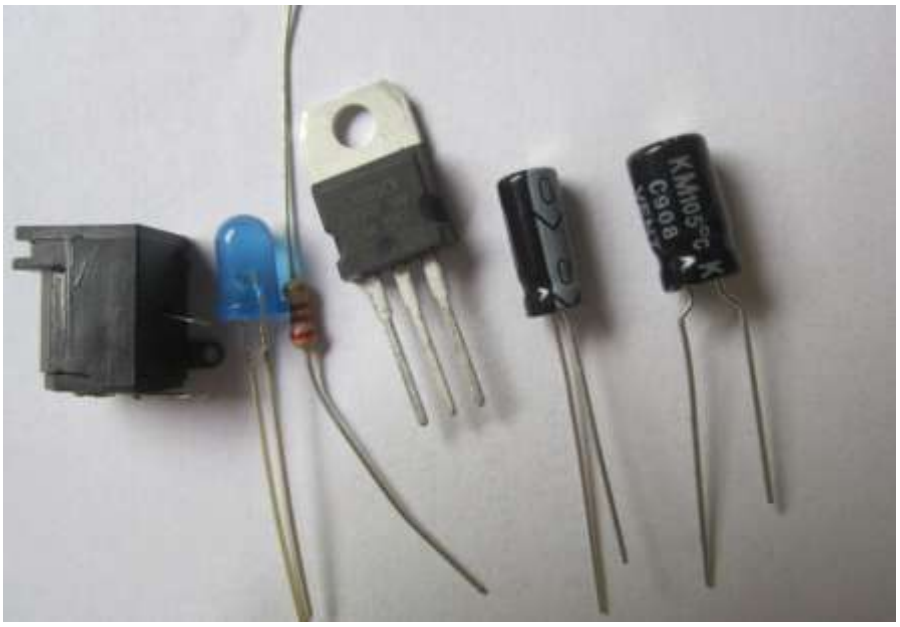

Figura 4.8- Dispositivos para circuito de alimentação

(FONTE: Autora)

4.2.1.5-Antenas

Para potencializar alcance da transmissão RF e assim podermos aumentar a distância entre o receptor e o dispositivo da mesa do cliente, necessitamos construir uma antena. A fórmula para o cálculo da antena e dada a seguir:

$$\lambda = \frac{c}{f} \quad (1)$$

Em que λ é o comprimento de onda, c é a velocidade da luz no vácuo (300.000.000 m/s) e f é a frequência da onda em Hertz, que neste caso é 433.92MHz.

Por este cálculo, definimos o comprimento de onda de $\lambda = 0{,}69137\text{m}$. Vários são os tipos de antena, algumas são helicoidais, outras isotrópicas ou $^1/_4$ de onda, etc Utilizando a antena $^1/_4$ de onda para o valor de λ encontrado, temos que o comprimento antena deve ser de 0,17284m ou 17, 2cm.

O alcance de um sinal de rádio frequência pode variar muito de acordo com o tipo de antena e do meio em que é utilizado. Se deixássemos de lado a fórmula, poderíamos utilizar

as disposições do fabricante que recomenda a utilização de antena do tipo vertical com os seguintes comprimentos:

• Transmissor TXC1: Comprimento = 22,6 cm para 315 MHz; Comprimento = 17,2 cm para 434 MHz (KEYMARK, 2002).

• Receptor RXD1: Comprimento = 22,6 cm para 315 MHz; Comprimento = 17,2 cm para 434 MHz (KEYMARK, 2002).

Uma vez que o projeto foi concebido para trabalhar com a frequência de 433.92 MHz, o tamanho da antena utilizada foi de 17,2 cm para o receptor e para o transmissor. A antena foi construída com fio de cobre encapado de 0,5 mm de diâmetro atingindo um comprimento de 17,2 cm. Essa antena é muito eficiente e possui fácil implementação. Tanto o *hardware* receptor como o *hardware* transmissor possui uma antena de fio de cobre com tamanho de 17,2 cm.

4.2.1.6- Montagem do circuito receptor

Os componentes do módulo receptor foram primeiramente conectados em uma *protoboard* para desenvolvimento do *firmware* e testes. A interligação dos circuitos pode ser conferida na figura abaixo.

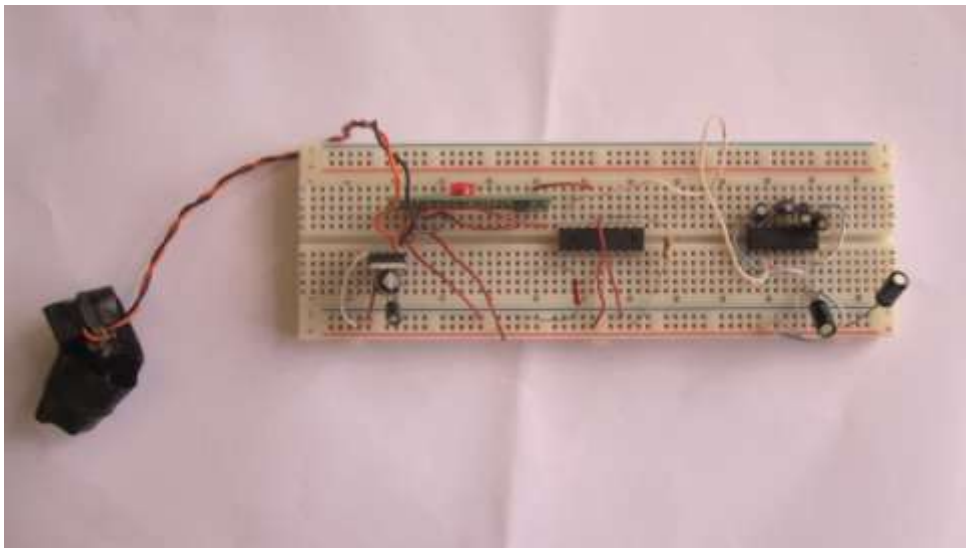

Figura 4.9- *Hardware* Receptor

(FONTE: Autora)

Com o protótipo testado em *protoboard*, o próximo passo foi a montagem do circuito em uma placa de fenolite perfurada. Ver figura 4.10

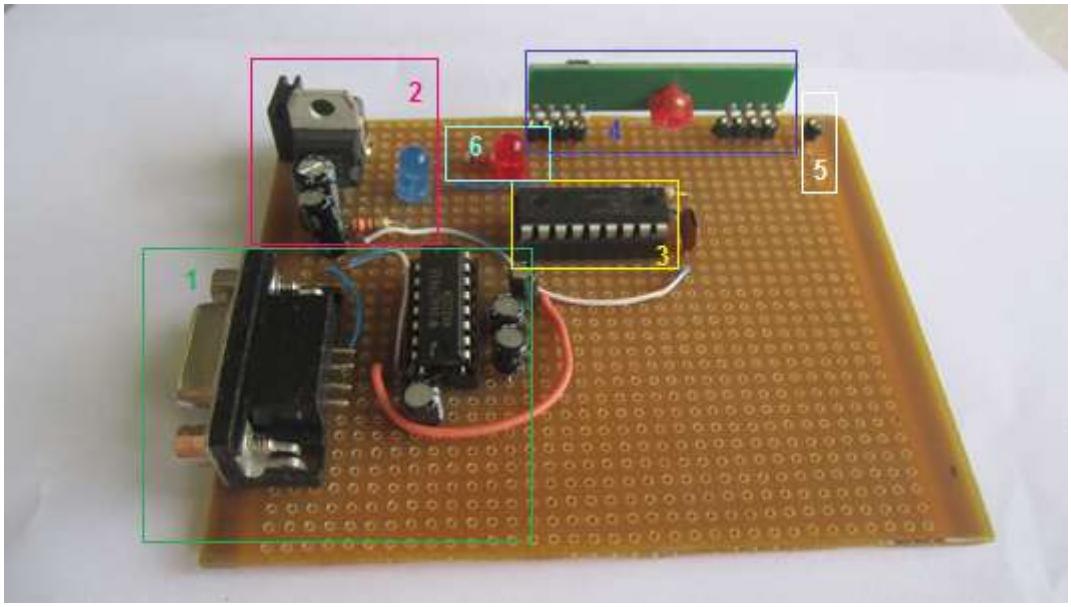

Figura 4.10- Placa do circuito receptor

(FONTE: Autora)

Na figura acima, são enumerados os cinco itens que compõem a placa. São eles:

1. Circuito MAX232 e conector DB9 fêmea;
2. Conector Jack para fonte externa e circuito LM7805
3. PIC16F628A;
4. Receptor RF RXD1;
5. Conector da antena
6. LED indicador de recepção de dados.

4.2.2- *Hardware* Módulo Transmissor

Na montagem deste módulo, três componentes principais foram utilizados para realizar a tarefa. São eles:

1. Botão para chamar o garçom
2. Microcontrolador PIC16F628A;
3. Transmissor RF TXC1 433MHz.

4.2.2.1-Botão para chamar o garçom

Este botão é acoplado ao *hardware* transmissor e assim que for pressionado, fará com que seja executada uma ação no microcontrolador, que terá sua tensão alterada.

Os resistores conectados ao botão são do tipo resistores de *pull-up / pull-down*. São resistores adicionados ao projeto para assegurar um certo nível lógico em um pino de entrada lógica, em outras palavras, eles forçam um estado (0 ou 1) no pino do microcontrolador.

Resistores de *pull- down* possuem praticamente o mesmo modo de operação, diferenciando dos de *pull-up* apenas pelo fato de serem conectados ao terra para forçar nível zero enquanto um botão (que agora será conectado ao Vcc) não é pressionado. Ver figura 4.11

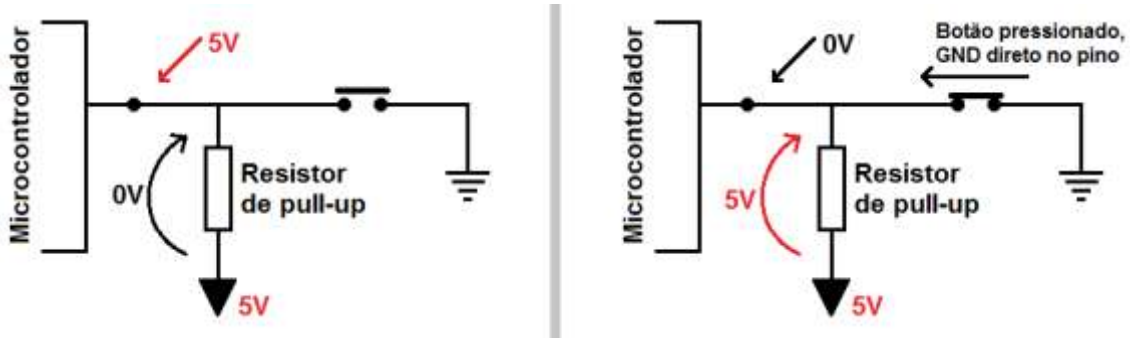

Figura 4.11- Resistores *pull-up*, quando o botão não está pressionado(esquerda) e quando está pressionado (direita)

(FONTE: Disponível em: < http://gustavolaureano.blogspot.com/2011/07/conceitos-basicos-parte-1-*pull-up*-pull.html>)

O resistores escolhidos foram de 10K *ohms*. Foi preciso cuidado na escolha do valor do resistor pois o seu valor não poderia ser baixo o suficiente para consumir uma corrente excessiva da fonte quando ele fosse aterrado, e nem alto o suficiente para que a corrente consumida pela entrada do microcontrolador (baixa, mas existente) causasse uma queda de tensão muito grande sobre o resistor, levando a entrada a ficar em um nível flutuante.

Outro fator importante na análise foi a resistividade de contato do botão utilizado. Se o botão apresentar uma resistividade alta (contato sujo ou oxidado pelo tempo) e o resistor terá um valor relativamente baixo e poderá haver na entrada uma tensão baixa o suficiente para causar flutuações, devido ao divisor de tensão formado entre o resistor e o botão.

4.2.2.2- Microcontrolador PIC16F628A

O microcontrolador PIC foi ligado de forma a receber o terra assim que o botão for pressionado, levando seu nível lógico para 0. O microcontrolador processa essa informação e encaminha os dados ao transmissor de radiofrequência. Nele também foram conectados um LED indicador, além do resistor de 10KΩ e capacitor de 1pF recomendados pela Microchip Inc.

Os pinos do microcontrolador ficaram conectados da seguinte forma:

- Pino 4 – Resistor 10KΩ e 5V;
- Pino 5 – Terra;
- Pino 8 – Pino 2 do TXC1;
- Pino 12 – LED verde indicativo de transmissão e pino 3 do TXC1;
- Pino 14 – 5V
- Pino 10- Botão push-button e resistor 10KΩ aterrado

O pino 4 do PIC é para *reset* do microcontrolador. O resistor conectado a ele é do tipo *pull-up*, que eleva a tensão do pino ao nível alto (5V) evitando que ocorram *resets* inesperados.

Os pino 5 e 14 são de alimentação do microcontrolador.

O pino de recepção de dados (RX) do microcontrolador é o pino 7

Já o pino 8 é o TX do microcontrolador e está conectado ao pino 2 (RX) do transmissor de radiofrequência.

Um LED foi ligado em série com um resistor de 220Ω ao pino 12 do microcontrolador e é utilizado para indicar quando há transmissão pelo módulo. Este pino também alimenta o transmissor TXC1.

No pino 10 estão conectados o botão *push-button* e resistor de 10 KΩ para que assim que o botão for pressionado o microcontrolador comece a executar suas instruções.

4.2.2.3- Transmissor TXC1

Este componente é responsável pela transmissão dos dados referentes ao número da mesa que está solicitando atendimento. Utiliza ondas de rádio que operam na faixa de frequência de 433.92 MHz. Na Figura 4.12 é apresentada a dimensão física do transmissor e a

especificação técnica de seus pinos. Do ponto de vista da transmissão de dados, dos quatro pinos existentes os mais importantes são os pinos 4(ANT) e 2(DATA), respectivamente a saída para a antena e a porta de entrada de dados. Os outros pinos estão ligados à alimentação do componente.

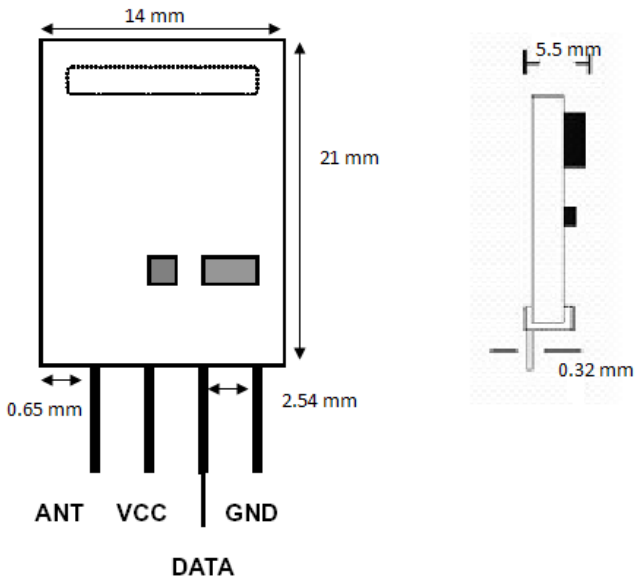

Figura 4.12 – Dimensões Físicas do Transmissor (Adaptado de KEYMARK, 2002)

Após processamento, as informações com o número da mesa são enviadas ao transmissor TXC1, que realiza a modulação do sinal e o envia para o receptor por ondas com frequência de 433MHz. Estes dados estão disponíveis para qualquer receptor que capte esta frequência, porém somente podem ser interpretadas por dispositivos que conheçam o protocolo de comunicação implementado.

A configuração dos pinos ficou da seguinte maneira:

- Pino 1 – Terra;
- Pino 2 – Pino 8 do PIC;
- Pino 3 – Pino 9 do PIC;
- Pino 4 – Antena.

O segundo pino do TXC1 é o de recepção de dados, e por ele o microcontrolador envia as informações a serem moduladas e transmitidas em RF.

Para que o TXC1 não transmita dados aleatórios enquanto não estiver em uso, a configuração foi feita para que o microcontrolador controle o momento de ligar e desligar o transmissor, para isso o pino de alimentação (3) foi ligado ao pino 9 do PIC.

4.2.2.4- Alimentação do circuito transmissor

A alimentação do *hardware* transmissor se deu por meio de uma bateria de 9V.

A tensão máxima suportada por todos os componentes é de 5V, para o ajuste da tensão para o nível correto, um regulador de tensão foi montado. O regulador utilizado foi o LM78L05 e ligado de acordo com *datasheet* do fabricante. A pinagem do regulador ficou com a seguinte configuração:

- Pino 1 recebe o positivo da bateria;
- Pino 2 é ligado ao negativo da bateria;
- Pino 3 é a saída ajustada a 5V.

Também é ligado entre os pinos 1 – 2 e 2 – 3 dois capacitores cerâmicos de 10pF, como recomendação do fabricante.

A Figura 4.3 ilustra o esquemático do circuito desenvolvido no Proteus.

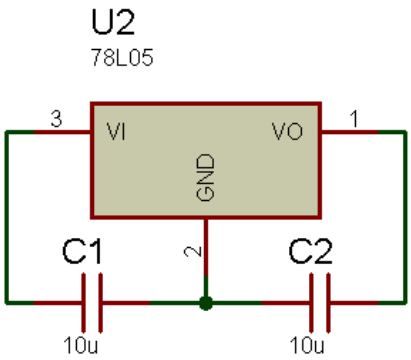

Figura 4.1**3** - Circuito LM78L05

(FONTE: Autora)

4.2.2.5- Montagem do circuito transmissor

Antes de os componentes do circuito transmissor serem soldados em uma placa de fenolite, foi desenvolvido primeiro o seu protótipo em uma *protoboard*, o que pode ser visto na figura 4.14. Feito os devidos testes, o próximo passo foi a confecção do circuito na placa de fenolite.

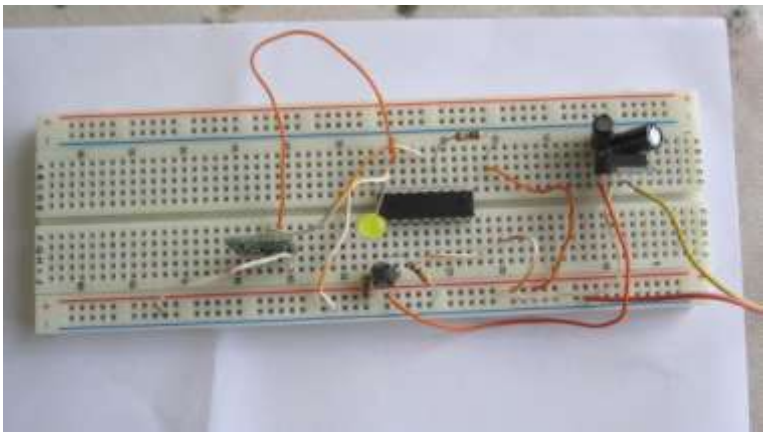

Figura 4.14- *Hardware* de Transmissão. (FONTE: Autora)

Na figura 4.15 são enumerados os cinco itens que compõem a placa. São eles:

1. Bateria 9 V
2. PIC16F628A;
3. Transmissor TXC1;
4. Botão
5. Led indicador de transmissão

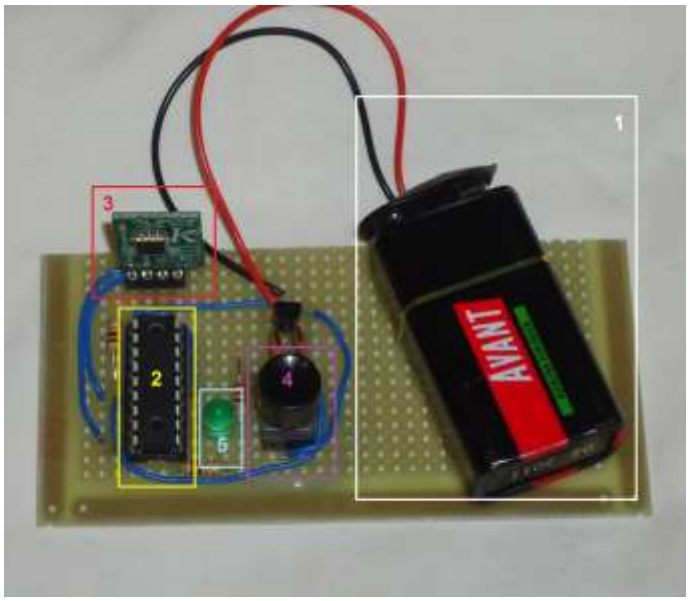

Figura 4.15 - Placa com circuitos enumerados
(FONTE: Autora)

Todas as configurações desse *hardware* foram utilizadas para a montagem do segundo *hardware* transmissor, uma vez que foi proposto a confecção de dois dispositivos de mesa. A única diferença entre os 2(dois) *hardware*s transmissores é que cada um está carregado com um número de mesa diferente. Todavia, essa configuração é feita apenas no firmware do microcontrolador. Por tudo isso, acredita-se que seja redundante a colocação da montagem do segundo *hardware* transmissor.

4.3- *Software*

Além do *hardware*, foi de fundamental importância o desenvolvimento da parte lógica do projeto. Pode-se dizer que dois *software*s foram implementados, o firmware dos microcontroladores e um aplicativo desktop utilizado para a visualização da solução e gerenciamento do sistema.

Firmware é o código gravado e executado no microcontrolador, é entendido como as instruções operacionais programadas diretamente no *hardware*.
O aplicativo desktop trata-se da interface gráfica, ou GUI (*Graphical User Interface*), que permite a visualização da organização das mesas do ambiente, permite ver qual mesa solicita o atendimento e a hora que o mesmo ocorreu, etc.

4.3.1 – Firmwares dos Microcontroladores

Para o firmware dos microcontroladores foi utilizada a linguagem C. Ainda que o código fonte final fique mais extenso, ocupando um espaço maior na memória, optou-se por esta linguagem devido à maior facilidade de implementação do código fonte.

A principal função dos microcontroladores é esperar a chegada de dados e então realizar algum procedimento específico. O microcontrolador do transmissor espera até que o cliente aperte o botão para então enviar as informações para o módulo transmissor e o microcontrolador do *hardware* receptor aguarda por essas informações que serão transmitidas via RF e que primeiro serão passadas ao receptor e depois encaminhadas a ele.

Em ambos os casos foram inseridos bytes de controle durante a transmissão para indicar início e fim da mesma.

O protocolo é um conjunto de regras e procedimentos que foram implementados para enviar e receber os dados de forma correta e para isso, a mensagem transmitida precisa seguir o padrão ilustrado na Tabela 4.1.

Tabela 4.1 – Protocolo de Comunicação
(FONTE: Autor)

Início da Mensagem	Endereço do Emissor	Endereço do Receptor	FIM
1 Byte	1 Byte	1 Byte	1 Byte

Para facilitar o entendimento do funcionamento dos *software*s foram descritos de acordo com o fluxo do sistema, primeiramente o processo de transmissão e então o processo de recepção. Os fluxogramas idealizam o funcionamento dos firmwares dos dispositivos de transmissão e recepção. A Figura 4.16 ilustra o *software* do dispositivo de transmissão.

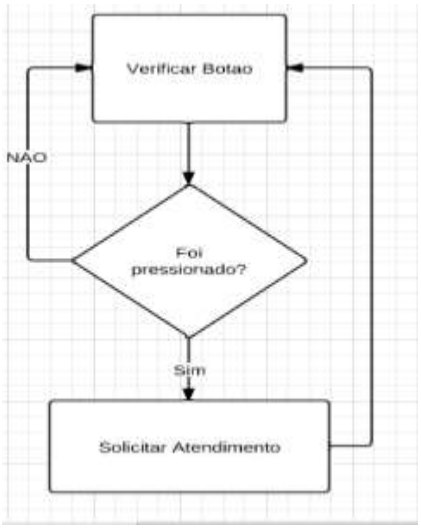

Figura 4.16 - Fluxograma da transmissão dos dados

(FONTE: Autora)

Neste programa primeiramente são estabelecidas as configurações para gravação no microcontrolador, como a definição do microcontrolador utilizado, a taxa de transmissão e a utilização das bibliotecas adequadas. Depois é realizada a definição e inicialização das variáveis e constantes internas. As entradas e saídas são configuradas e, em seguida, é realizada a definição das funções utilizadas no programa. O botão push button quando acionado faz com que o programa principal chame a função solicitar_atendimento, esta por sua vez quando é inicializada, liga o transmissor. São necessários 50 milisegundos para que ocorra a estabilização do sistema, logo depois bits de sincronismo são enviados antes do bit de INICIO. Este bit é seguido pelo endereço do receptor, pelo número da mesa que está solicitando o garçom e por fim pelo bit que sinaliza o fim da mensagem . Para finalizar a função solicitar_atendimento desliga o transmissor.

Já o fluxograma do *software* receptor pode ser conferido na Figura 4.17.

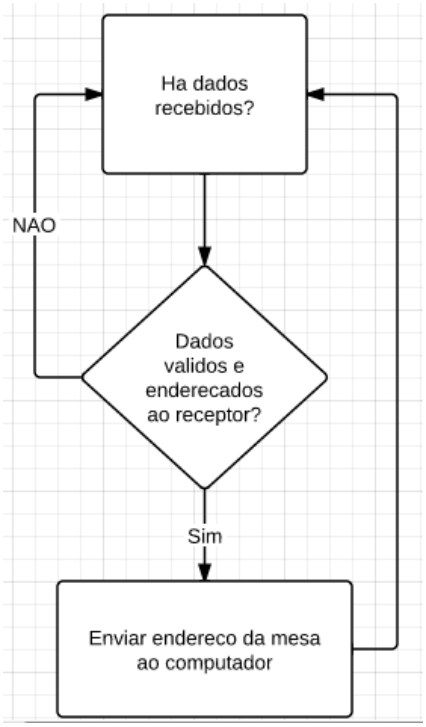

Figura 4.17 - Fluxograma do receptor

(FONTE: Autora)

De modo semelhante ao que ocorre no *software* do transmissor, aqui primeiramente são estabelecidas as configurações para gravação no microcontrolador. Depois é realizada a definição e inicialização das variáveis e constantes internas. As entradas e saídas são configuradas e, em seguida, é realizada a definição das funções utilizadas no programa.

4.3.2– Interface Gráfica do Usuário

A interface gráfica do usuário foi desenvolvida a fim de facilitar a interação do usuário final com os dispositivos. A interface é composta por poucas classes, são elas: GarcomDigital, Config, e Monitor. Com exceção da classe GarcomDigital, todas as outras possuem algumas funções já prontas como é o caso da função Clock por exemplo que permite a visualização da hora atual na tela do sistema e registra a hora certa em que uma mesa solicitou o atendimento. Por meio desta função clock, o usuário final pode identificar a ordem em que as mesas solicitaram atendimento, entretanto é interessante que seja implementada uma forma de visualização maior, para que os garçons possam ter essa informação assim que olharem para tela, mesmo que estejam distantes do microcomputador.

Pensando nisso, o programa mostra a primeira mesa que solicitou o pedido com a cor do realce vermelha e a segunda mesa com a cor do realce verde.

Além disso, todo o programa possui partes que não podem ser alteradas. São informações criadas automaticamente pelo Swing assim que foram adicionados objetos(botões, figuras, entre outros) na tela .

O Java possui interfaces de escuta chamados *Listeners*. Estas interfaces são protótipos de funções que quando utilizadas, monitoram a execução dos componentes. Cada botão da interface gráfica possui um *Listener* chamado *ActionListener*, que monitora quando uma ação é executada no botão, como por exemplo o clique do mouse, e a partir disto executa a função destinada àquele botão.

A classe GarcomDigital possui 2 botões(Jbutton), o 'Monitorar Mesas' e o 'Fechar' como é ilustrado na figura 4.18. O 'Fechar' encerra a aplicação já o Jbutton 'Monitorar Mesas' quando acionado, instancia a classe Config e a classe Monitor

Figura 4.18 – Interface inicial do programa

(FONTE: Autora)

Quando executada, a classe Config exibe ao usuário uma janela com dois botões. Esta nova janela, exibe as opções de portas seriais disponíveis e as velocidades de transmissão em dois componentes Java denominados *JComboBox*. O botão "Conectar" dá continuidade à execução da janela, permitindo que a conexão serial seja estabelecida a partir das opções selecionadas nos combos. Já o "Fechar" interrompe a ação de conectar ao transmissor. A Figura 4.19 ilustra a janela de configuração da porta serial.

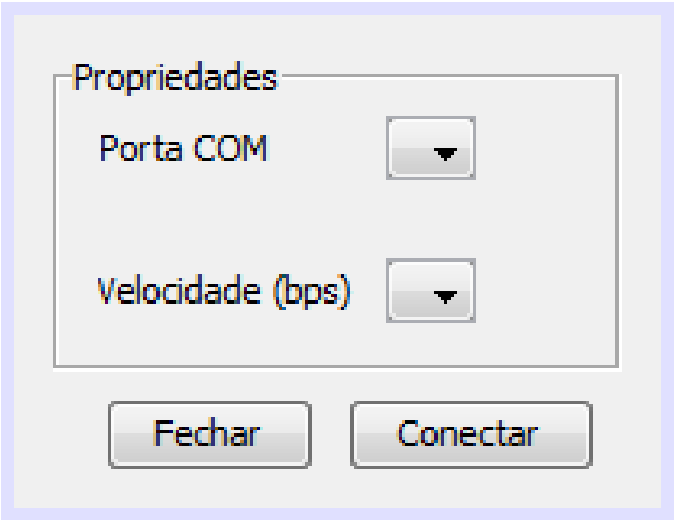

Figura 4.19 - Janela de configuração da porta serial

(FONTE: Autora)

A classe Monitor é a responsável pela visualização da organização das mesas, relação de horários e mesas que efetuaram o chamado. Ela possui um painel que contém 12(doze) JButton e 12(doze) JTextField organizados em 3(três) fileiras, cada uma com 4(quatro) colunas , essa organização simula a disposição das mesas do ambiente. Esta classe possui também um JTextFiel Relógio para monitorar os horários de atendimento com precisão. Além de uma JTextArea onde aparece o histórico das mesas atendidas e um JButton “Parar Garçom Digital” para encerrar a aplicação. É importante mencionar que para a visualização de todas as funcionalidades desta classe, todos os dispositivos precisam estar devidamente conectados, principalmente o *hardware* receptor deve estar ligado ao computador onde se quer visualizar o programa. A Figura 4.20 ilustra as informações referidas acima.

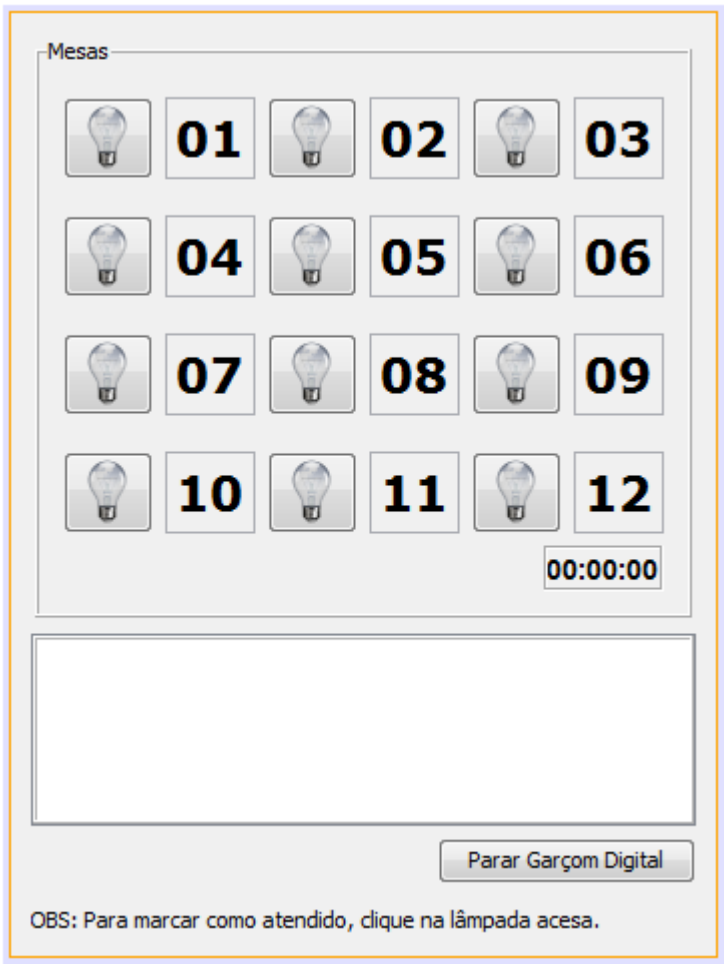

Figura 4.20 - Tela principal do programa

(FONTE: Autor)

CAPÍTULO 5- APLICAÇÃO DO MODELO PROPOSTO

Este capítulo apresenta as simulações que foram realizadas no projeto a fim de evitar queima de equipamentos e otimizar os componentes utilizados. Além disso, traz o passo-a passo para a utilização do projeto bem como as dificuldades enfrentadas para a sua realização e ainda o custo estimado gasto para a construção do sistema apresentado neste trabalho de conclusão de curso.

5.1- Simulações

Antes da montagem do protótipo físico foram realizadas simulações no programa Proteus. Conforme pode ser observado nas Figuras 5.1 e 5.2, o dispositivo das mesas e o *hardware* receptor foram montados com todos os componentes necessários para o funcionamento do *hardware*, como se estivesse sendo montado com os componentes físicos reais. A utilização do *software* de simulação foi de extrema importância para a realização do projeto, pois possibilitou identificar as falhas da utilização de componentes e, principalmente, para debugar os programas para transmissão, recepção e controle dos periféricos. Os programas, do dispositivo das mesas e do dispositivo receptor foram embarcados nos respectivos microcontroladores e foram realizados os ajustes necessários.

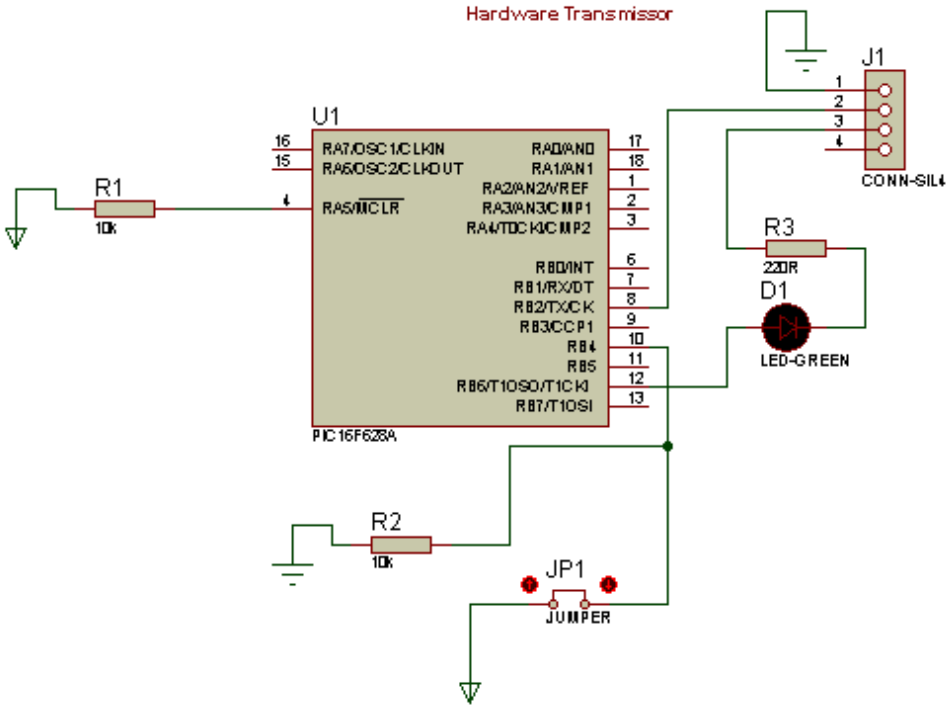

Figura 5.1 - Simulação do circuito elétrico *Hardware* Receptor

(FONTE: Autora)

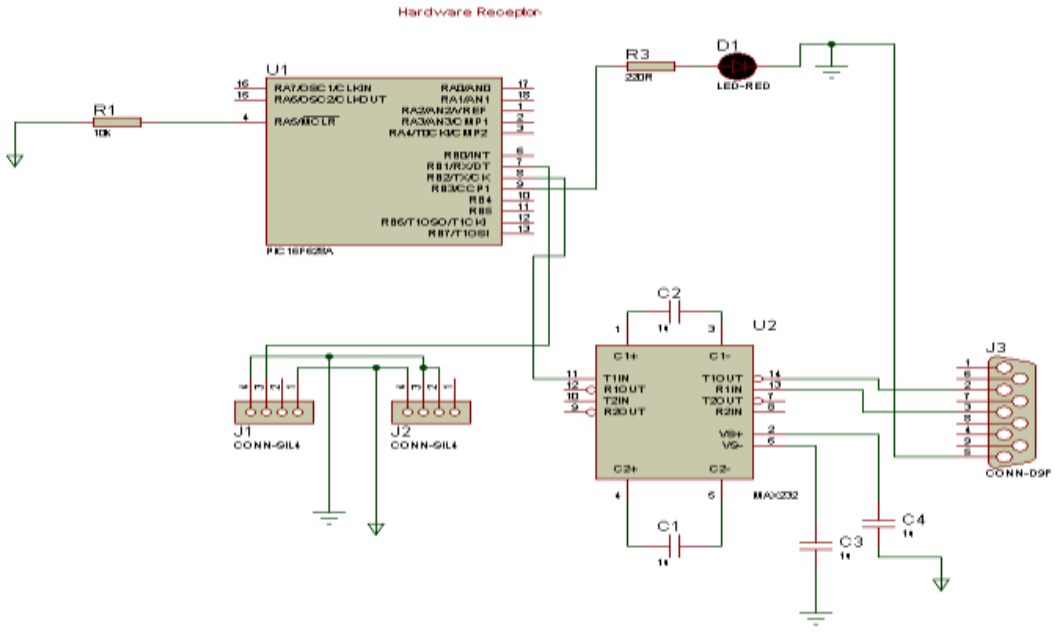

Figura 5.2- Simulação do circuito elétrico *Hardware* Transmissor

(FONTE: Autora)

Por uma limitação da ferramenta Proteus, que não possibilita a simulação de transmissão sem fio entre os dispositivos, os testes foram realizados com a comunicação ponto a ponto utilizando um barramento direto da porta (**RB2/TX/CK**) do microcontrolador do dispositivo das mesas para a porta (**RB1/RX/DT**) do micro-controlador do dispositivo receptor. As simulações foram satisfatórias e viabilizaram a construção do *hardware* fisicamente como pode ser visto na figura 5.3.

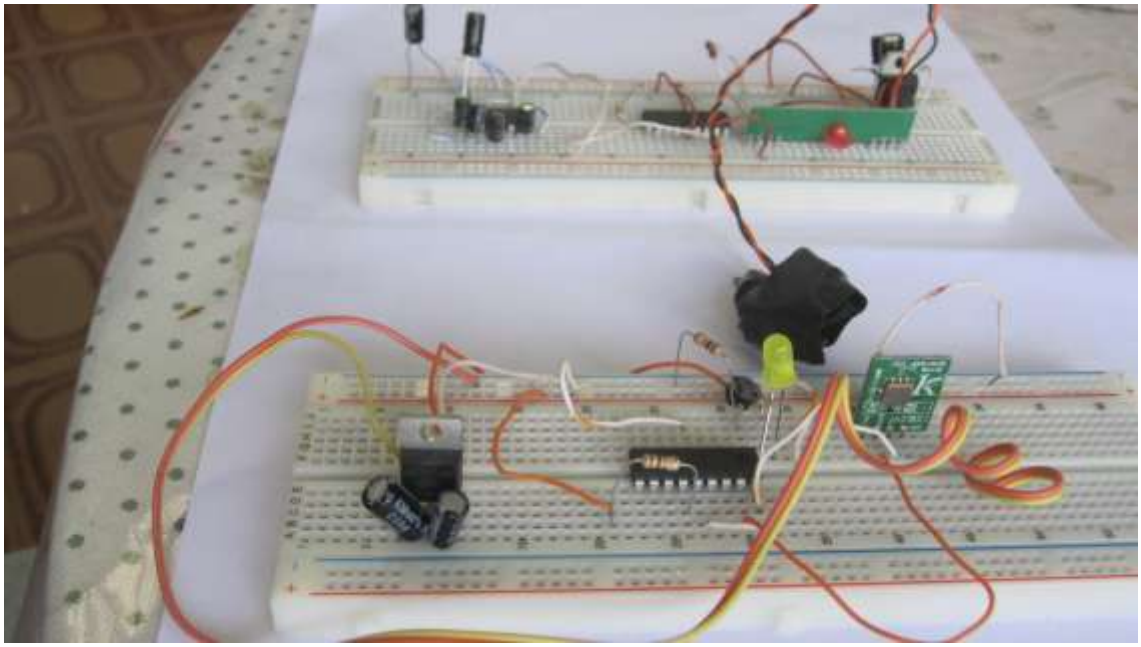

Figura 5.3- Garçom Digital na *Protoboard*

(FONTE: Autora)

5.2- Testes

Levando-se em conta os objetivos propostos iniciaram-se os testes. Para começar foram conectados todos os equipamentos e testada a comunicação entre os dispositivos, conforme a seqüência de ações abaixo:

1ªEtapa: Conexão de todos os equipamentos

- Placa do circuito transmissor devidamente conectada à bateria de 9V.
- Placa do circuito receptor conectada ao computador (Modelo Sony VPCEA33FB);
- Fonte conectada à placa do circuito receptor e ligada à tomada (220 V);
- Computador ligado à tomada (220 V).
- Led vermelho piscou e Led azul permanece acesso indicando a passagem de corrente na placa do circuito;

A figura 5.4 ilustra esta etapa em andamento. Na figura ainda falta conectar o computador e a fonte de 16.5 V.

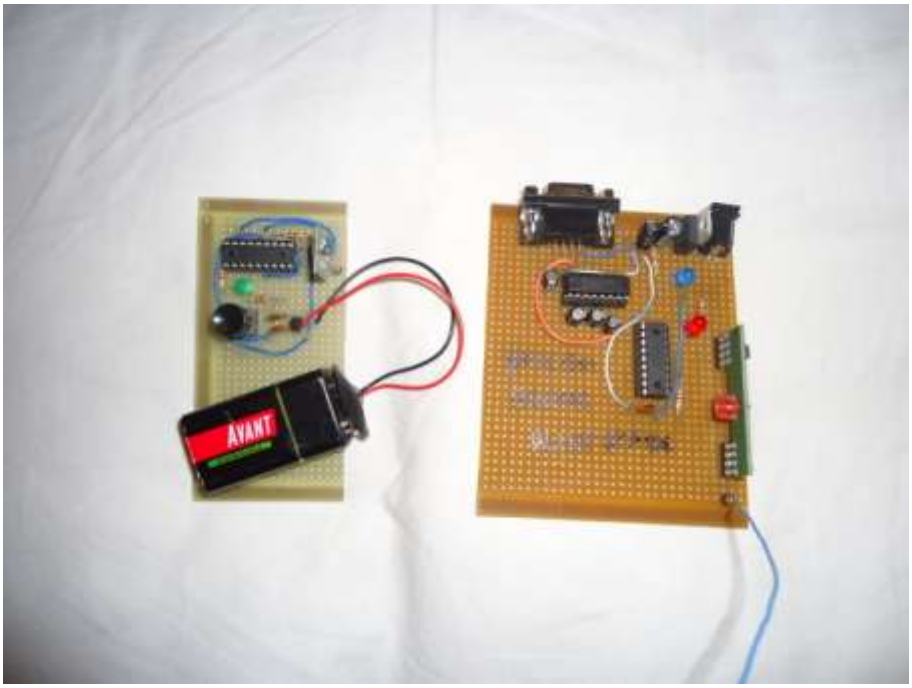

Figura 5. 4 - Placas dos circuitos

(FONTE: Autora)

2ª Etapa: Ativação do *software* garçom digital instalado no computador;

- Duplo-click no arquivo GarcomDigital.jar

-Na tela seguinte clique simples no botão MONITORAR MESAS

3º Etapa Conexão do aplicativo de gerência(*software* garcomdigital) à porta serial padrão, conforme Figura 5.5;

- Escolha da porta COM e da taxa de transmissão (Baud Rate) 1200 bps

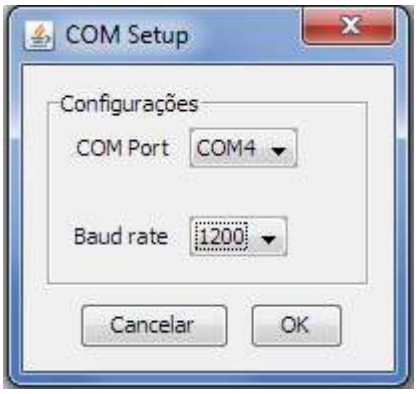

Figura 5. 5 – Escolha da porta COM e baud rate

(FONTE: Autora)

4º Visualização da organização das mesas do restaurante

- Nessa tela seguinte que aparece é possível verificar a disposição das mesas do restaurante. A figura 5.6 ilustra o resultado do chamado das mesas.

Em seguida, foi executado o teste para a verificação do correto funcionamento do sistema. Deste modo, foi possível avaliar se o número que aparecia em realce na tela do computador correspondia ao número da mesa que teve o botão do seu *hardware* acionado. Por meio deste teste foi possível validar o correto funcionamento dos dispositivos de *hardware* e a correta transmissão dos números das mesas. A figura 5.6 ilustra o layout das mesas dos restaurantes e a aparência em destaque do número da mesa que efetuou o chamado.

A primeira mesa que solicitou atendimento assume a cor vermelha e a segunda mesa assume a cor verde. Essa solução só é viável para ambientes pequenos e com poucas mesas. Para ambientes maiores, para fazer a ordenação das mesas, uma solução, seria criar um painel que exibisse a fila de mesas que querem ser atendidas.

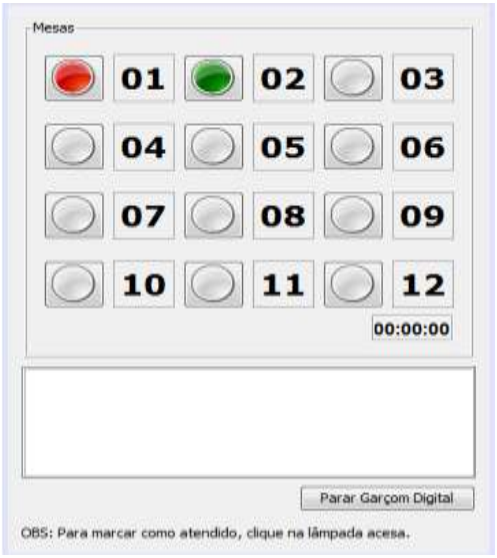

Figura 5. 6 – Painel com mesas solicitando atendimento

(FONTE: Autora)

Outro teste realizado foi a alteração da velocidade da transmissão. Como os *datasheet* dos módulos de transmissão/recepção utilizados neste projeto não possuem especificação da velocidade de transmissão, decidiu-se fazer esse teste com velocidades usuais encontradas na comunidade Web de módulos semelhantes ao utilizados no projeto.

Como a taxa de transmissão de 1200bps já havia sido testada como demonstrado há pouco, agora o teste foi realizado com 2400bps. Para esta velocidade, o teste mostrou que a transmissão ocorria com falhas e nem sempre o número da mesa ficava em realce na tela do computador.

5.3– Dificuldades

Ainda que os componentes utilizados na confecção deste projeto sejam simples e teoricamente poderiam ser encontrados em qualquer loja de eletrônica, alguns dispositivos como os módulos híbridos de radiofrequencia foram de difícil acesso pois estavam em falta nas principais lojas do Distrito Federal e o pedido pela internet sofreu atrasos devido a greve dos correios que ocorreu no período de 14 de Setembro a 11 de Outubro.

Antes da correta implementação de um protocolo de comunicação entre os dispositivos de transmissão e recepção, a comunicação também apresentou dificuldades para realizar a sincronização de recebimento de pacotes pelo receptor.

5.4– Custo do Projeto

A tabela 5.1 representa uma estimativa do investimento efetuado no projeto.

Tabela 5.1 – Valor Estimado do Projeto

Quantidade	**Nome**	**Preço(R$)**
3	PIC16F628A	30,00
1	Notebook Sony	2000,00
2	Receptor RF	16,00
2	Transmissor RF	30,00
1	Placa de Fenolite 20x10	16,00
1	Cabo conector DB9	28,00
	Demais componentes	40,00
Total		2160,00

FONTE: Autora

Considerando os valores mostrados na tabela 5.1, pode-se perceber que o item Notebook Sony possui preço bem elevado se comparado aos demais itens registrados na tabela. Como a autora do projeto já possuía este item afirma-se que o gasto estimado do projeto foi de aproximadamente R$ 160,00 (cento e sessenta reais).

CAPÍTULO 6- CONSIDERAÇÕES FINAIS

6.1 – Conclusões

Este projeto teve como objetivo criar um sistema que melhorasse a qualidade do atendimento ao cliente em estabelecimentos de alimentação. Pensando nisso desenvolveu-se um sistema que oferece a possibilidade de uma pessoa chamar o garçom até a sua mesa com um simples apertar de botão.

Quando este botão foi acionado, o microcontrolador PIC16F628A comunicou via transmissão de radiofrequência o número da mesa que estava solicitando atendimento a um receptor RF conectado a outro PIC16F628A. Este por sua vez encaminhou as informações para o conversor MAX232 para que o número da mesa pudesse ser exibido em realce em uma tela de um microcomputador.

Apesar de todos os problemas encontrados, o objetivo do trabalho foi alcançado. O projeto é capaz de transmitir o número de uma determinada mesa a outro dispositivo utilizando como meio de transmissão ondas de radiofrequência.

Com o apoio dos conhecimentos obtidos em diversas disciplinas do curso de Engenharia da Computação, das pesquisas bibliográficas e dos testes realizados foi possível construir um dispositivo estável, capaz de cumprir com a tarefa proposta. Os resultados foram atingidos com sucesso, pois o correto número de cada mesa pôde ser visualizado na simulação da organização do ambiente por meio da tela do monitor.

Além disso, o desenvolvimento ajudou a reforçar, na pratica, assuntos diversos vistos em sala de aula e, também possibilitou a aquisição de novos conhecimentos. A citar pode-se falar do aprendizado do modo de funcionamento do circuito MAX232 e da programação de um aplicativo com interface gráfica usando a linguagem JAVA

O presente projeto é apresentado como uma oportunidade de incorporar novos conhecimentos sobre tecnologia recentes, muito utilizadas no mercado das automações e outros projetos. Este mercado está sempre em expansão devido ao grande volume de possibilidades a serem exploradas, ainda mais pela vinda de eventos esportivos de grande porte para o Brasil, que faz com que estabelecimentos do ramo alimentício queiram cada vez mais agilizarem o atendimento e melhorarem a qualidade do serviço prestado.

6.2 - Sugestões Para Trabalhos Futuros

Durante os estudos e pesquisas para o desenvolvimento deste trabalho surgiram algumas ideias para projetos futuros. São elas:

-Implementação no *software* de uma função que gere o relatório de atendimento das mesas indicando qual garçom atendeu a mesa, qual o tempo médio de espera para o atendimento e quanto a mesa consumiu. O armazenamento dessas informações possibilitaria ao garçom ter o controle de quanto deverá receber de gorjeta e ao estabelecimento ter o monitoramento das atividades do garçom.

-Desenvolvimento de uma aplicação para as mesas dos clientes que possibilitasse a ele fazer o pedido sem a presença de um garçom, este dispositivo seria uma espécie de cardápio eletrônico.

-Implementação de um dispositivo para o garçom que permitisse a retirada do realce das mesas para que dessa forma, o garçom não tivesse que se locomover até o computador para efetuar a retirada

REFERÊNCIAS BIBLIOGRÁFICAS

AHO, A V.[et.al] .Compiladores: princípios, técnicas e ferramentas/ Alfred V. Aho....[et al.]; Tradução Daniel Vieira; revisão técnica Mariza Bigonha.- 2 ed. São Paulo: Pearson Addson-Wesley, 2008.

ALBRECHT, K: Trazendo o poder do cliente para dentro da empresa: a únicacoisa que importa, São Paulo : Pioneira, 1995.

BYRD, T.A. & MARSHALL, T.T.: "Relating Information Technology Investment to Organizational Performance: a Causal Model Analysis". Omega, International Journal of Management Science, v.25, n.1, p.43-56, 1997.

CASTELLI, G.: Administração hoteleira , Caxias do Sul EDUCS 2001

CENTRO UNIVERSITARIO NOVE DE JULHO. Regulamento de TCC – Trabalho de conclusão de curso. São Paulo, Coordenação de Estágio Curricular – UNINOVE- SP/2001

COBRA, M. Serviços ao cliente: uma estratégia competitiva. 2ª ed., SãoPaulo Marcos Cobra Editora, 1993

DEITEL, P. J.; DEITEL, Harvey M.. Java TM: como programar. Tradutor: Carlos Arthur Lang Lisbôa. 4ª. ed. Porto Alegre, RS: Bookman, 2003. 1201 p. il. ISBN 85-7307-727-1.

DEITEL, H. M.; DEITEL, P. J. Java.Como programar. 6. ed. Prentice Hall Brasil, 2006.

DUFFY, D L..: Do something ! : guia prático para fidelização de clientes,tradução de Frank Edwin Duuvoort, São Paulo Prentice Hall 2003

GIGLIO, E. O comportamento do consumidor e gerência de marketing. SãoPaulo: Pioneira, 1996

JAMSA, K; KLANDER, Lars. Programando em C/C++ A Biblia. Editora afiliada- Sao Paulo 1999. Páginas: 1012.

KEYMARK Technology. Data Sheet ASK Transmitter Module. USA 2002

KEYMARK Technology. Data Sheet ASK Receiver Module. USA. 2003

LUFTMAN, J.N.; LEWIS, P.R. & OLDACH, S.H.: "Transforming The Enterprise: The Alignment Of Business And Information Technology Strategies". IBM Systems Journal, v.32, n.1, p.198-221, 1993.

MARTINS, N A. Sistemas Microcontrolados. 1Ed. São Paulo: Novatec, 2005.

MARICATO, P.: Como montar e administrar bares e restaurantes : um guia para atuais e futuros empresários do setor , Rio de Janeiro TQC 1997

_____. A excelência no atendimento ao cliente.2003. Disponível em http://www.tqceditora.com.br/artigos.asp?categoria=ATENDIMENTO. Acessado em Agosto de 2011

Microchip. (2007). datasheet PIC16F628A.

PRESSMAN, R. S. Engenharia de *software*. 6. ed. São Paulo: Mc Graw Hill Interamericana do Brasil, 2006

SOARES, L.F.G & LEMOS, Guido & COLCHER, Sérgio. Redes de Computadores – Das Lans, Mans e Wans às Redes ATM. 2.Ed. Rio de Janeiro: Campus, 1995.

SOUZA, D. J,. **Desbravando o PIC**. Editora Erica, 2008.

PEREIRA, F. Microcontroladores PIC: Programação em C/Fábio Pereira. 7ed. São Paulo: Érica, 2007

PETERSEN , L. UART test program for 16F628. Montagem do Circuito MAX232. -2002. Disponível em:< http://www.oz1bxm.dk/PIC/628uart_c.htm>

SHIN CHIN INDUSTRIAL. (2010). Datasheet push-button switch – R13-507. Disponível em SCI Parts: http://www.sci.com.tw/PRODUCTS/switch/%28R13%29%20PUSH%20SWITCH/R13-507.htm. Acessado em 11 de Agosto de 2011,

TANENBAUM, A.S. REDES DE COMPUTADORES. Rio de Janeiro. Andrew S. Editora Campus, 1994.

WALTER, J. R, LUNDBERG Donald E. O restaurante: conceito e operação. 2003

ZANCO, W.S. Microcontroladores PIC 16F628A/648A: Uma abordagem Prática e Objetiva(1ª ed.). São Paulo. Érica

ZEMKE,R.; SCHAAF, D. A nova estratégia do marketing: atendimento ao cliente. São Paulo; Harbra, 1991.

Sites Consultados:

Artigos de Administração.Disponível em: http://www.artigonal.com/negocios-admin-artigos/o-cliente-em-primeiro-lugar-2236534.html. Acessado em 25/10/2011

Automação em Restaurantes. Disponível em < http://gestaoderestaurantes.wordpress.com/2008/05/04/pesquisa-sobre-automao-em-restaurantes/> Acessado em 25 de Setembro de 2011

Histórico Rádio Frequência. Disponível em < http://www.ancorador.com.br/negocios/telecomunicacoes/a-comunicacao-atraves-da-radio-frequencia-contextualizacao-historica> Acessado em 25 de Setembro de 2011

IBM. Disponível em: http://www.ibm.com/expressadvantage/br/articles_general_industry_3Q7.phtml. Acessado em 25/10/2011

Microcontroladores PIC. Disponível em < http://www.radioamadores.net/files/microcontroladores_pic.pdf> Acessado em 25 de Setembro de 2011

Monografia Satisfação do Cliente. Disponível em: <http://br.monografias.com/trabalhos3/satisfacao-cliente/satisfacao-cliente2.shtml> Acessado em 12 de Outubro de 2011

Sistema automatização bares e restaurantes. Disponível em < http://www.cin.ufpe.br/~if119/propostas/AER.htm> Acessado em 25 de Setembro de 2011

Sistemas de Controle com PIC e Porta Serial. Disponível em <http://armandokeller.com/blog/2010/05/sistema-de-controle-com-pic-e-porta-serial-rs232-parte-2/> Acessado em 12 de Outubro de 2011

Sistemas Digitais.Disponível em < http://www.cp.utfpr.edu.br/chiesse/Sistemas_Digitais/PIC16f628a.pdf> Acessado em 25 de Setembro de 2011

Referências ABNT. Disponível em <http://www.bu.ufsc.br/ccsm/vancouver.html> Acessado em 5 de Setembro de 2011

APÊNDICE

Apêndice A- Código Microcontrolador Hardware Receptor

```
#include <16f628a.h> //Inclui o cabeçalho específico do pic a ser utilizado
#use delay(clock=4000000) // Seta o clock interno para 4 Mhz
#fuses INTRC,NOWDT,PUT,NOBROWNOUT,NOLVP,NOMCLR

#use rs232(baud=1200, xmit=PIN_B2, rcv=PIN_B1,ERRORS) /* Seta o baud rate para
1200 e define o pino B2 como TX e B1 com RX*\

#define RECEPTOR  0x27
#define SYNC      0xF0
#define INICIO    0x01
#define FIM       0xFF

unsigned int8 datar;
int1 flag = 0;
int1 syn = 0;
int8 lock = 0;
int8 mesa;


#INT_RDA

void isr_rda() {//É  a função que é acionada quando ocorre a interrupção

  datar = getc();// Armazena o conteúdo que chegou na porta serial na variável datar

  if(datar == SYNC && lock == 0) {
    lock++;
  }
  else if(datar == INICIO && lock == 1) {
    lock++;
  }
  else if(datar == RECEPTOR && lock == 2) {
    lock = 0;
    syn = 1;
  }
  else if(datar == FIM && syn) {
    syn = 0;
    flag = 1;
  }
  else {
    lock = 0;
  }
  if(syn) {
    mesa = datar;
  }
}
```

```
void pisca_led() {
  output_high(pin_b3);
  delay_ms(150);
  output_low(pin_b3);
  delay_ms(150);
}

void main()   {

  delay_ms(150);

  setup_uart(1200);//Põe a velocidade da porta serial em 1200 bps

  enable_interrupts(INT_RDA);// Habilita interrupção da comunicação serial
  enable_interrupts(GLOBAL);//Habilita interrupção global

  delay_ms(2500);

  pisca_led();
  printf("Monitor ligado...\n\r");
  while(TRUE)                          // loop always
  {
    if(flag) {//Se chegou mensagem eu mando para o computador o endereço da mesa e
um byte para sinalizar o fim da mensagem

      putc(mesa);
      putc(0xFE);
      flag = 0;
    }

  }
}
```

Apêndice B- Código Microcontrolador Hardware Transmissor 1

```
#include <16f628a.h>
#use delay(clock=4000000)

#fuses INTRC_IO,NOWDT,PUT,BROWNOUT,NOLVP,NOMCLR

#use rs232(baud=1200, xmit=PIN_B2, rcv=PIN_B1,ERRORS)

#define INICIO 0x01

#define FIM 0xFF

#define RECEPTOR 0x27

#define MESA 0x48

#define SYNC 0xF0

//INICIO END RECEPTOR - MESA -  FIM

void solicitar_atendimento()

{

  output_high(pin_B6);    // Liga o transmissor

  delay_ms(50);           // Aguarda estabilizar

  putc(SYNC);             // Envia varios bits de sincronismo

  delay_us(250);

  putc(SYNC);

  delay_us(250);

  putc(SYNC);

  delay_us(250);

  putc(SYNC);

  delay_us(250);

  putc(SYNC);

  delay_us(250);

  putc(INICIO);           // Envia bit de INICIO da mensagem

  delay_us(250);

  putc(RECEPTOR);         // Envia o endereço do receptor
```

```
  delay_us(250);
  putc(MESA);            // Envia número da MESA que esta solicitando atendimento
  delay_us(250);
  putc(FIM);            // Envia o bit sinalizador de fim da mensagem
  delay_us(250);
  delay_ms(250);
  output_low(pin_B6);     // Desliga o transmissor
}
//Programa principal
void main()
{
  setup_uart(1200);
  delay_ms(500);
  output_high(pin_b6);
  delay_ms(500);
  output_low(pin_b6);
  while(TRUE) {
    while(!input(PIN_B4)) {}
    delay_ms(100);
    solicitar_atendimento();
    delay_ms(3000);
  }
}
```

Apêndice C- Código Microcontrolador Hardware Transmissor 2

```
#include <16f628a.h>
#use delay(clock=4000000)

#fuses INTRC_IO,NOWDT,PUT,BROWNOUT,NOLVP,NOMCLR

#use rs232(baud=1200, xmit=PIN_B2, rcv=PIN_B1,ERRORS)

#define INICIO 0x01

#define FIM 0xFF

#define RECEPTOR 0x27

#define MESA 0x51

#define SYNC 0xF0

//INICIO END RECEPTOR - MESA -  FIM

void solicitar_atendimento()

{

  output_high(pin_B6);    // Liga o transmissor

  delay_ms(50);           // Aguarda estabilizar

  putc(SYNC);             // Envia varios bits de sincronismo

  delay_us(250);

  putc(SYNC);

  delay_us(250);

  putc(SYNC);

  delay_us(250);

  putc(SYNC);

  delay_us(250);

  putc(SYNC);

  delay_us(250);

  putc(INICIO);           // Envia bit de INICIO da mensagem

  delay_us(250);

  putc(RECEPTOR);         // Envia o endereço do receptor

  delay_us(250);
```

```
  putc(MESA);            // Envia número da MESA que esta solicitando atendimento
  delay_us(250);
  putc(FIM);             // Envia o bit sinalizador de fim da mensagem
  delay_us(250);
  delay_ms(250);
  output_low(pin_B6);    // Desliga o transmissor
}

//Programa principal

void main()
{
  setup_uart(1200);
  delay_ms(500);
  output_high(pin_b6);
  delay_ms(500);
  output_low(pin_b6);

  while(TRUE) {
    while(!input(PIN_B4)) {}
    delay_ms(100);
    solicitar_atendimento();
    delay_ms(3000);
  }
}
```

Apêndice D- Código Garçom Digital

```
package garcomdigital;

import gnu.io.CommPortIdentifier;
import java.awt.Dimension;
import java.awt.Point;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;

/**
 *
 * @author Aline
 */
public class GarcomDigital extends javax.swing.JFrame {

    /** Creates new form GarcomDigital */
    public GarcomDigital() {
        initComponents();
        setTitle("Garçom Digital - por Aline Ataíde");
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = getSize();
        setLocation(new Point((screenSize.width - frameSize.width) / 2,
                        (screenSize.height - frameSize.width) / 2));
        ActionListener MonitorMesas = new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
              MonitorActionPerformed(e);
            }
        };
        ActionListener FecharJanela = new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
              FecharActionPerformed(e);
            }
        };

        Monitor.addActionListener(MonitorMesas);
        Fechar.addActionListener(FecharJanela);

    }

    private void MonitorActionPerformed(ActionEvent evt) {
        CommPortIdentifier portId;
        Integer baudRate;
```

```
        Config receptor = new Config(this);
        Monitor mesas = new Monitor(this);
        receptor.setVisible(true);
        baudRate = receptor.getBaudRate();
        portId = receptor.getSelectedIdentifier();
        if(baudRate != 0 && portId != null)
            mesas.conectar(portId, baudRate);
        if(mesas.conectado()) {
            setState(ICONIFIED);
            mesas.setVisible(true);
            if(!mesas.isVisible())
                setState(NORMAL);
        }
    }

    private void FecharActionPerformed(ActionEvent evt) {
        System.exit(0);
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        Monitor = new javax.swing.JButton();
        Fechar = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        Monitor.setText("Monitorar Mesas");

        Fechar.setText("Fechar");

        jLabel1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/waiter.png")));
// NOI18N

        jLabel2.setFont(new java.awt.Font("Verdana", 1, 24)); // NOI18N
        jLabel2.setText("Garçom");

        jLabel3.setFont(new java.awt.Font("Verdana", 1, 24)); // NOI18N
        jLabel3.setText("Digital");

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                    .addComponent(jLabel1)
                    .addComponent(jLabel2)
                    .addComponent(jLabel3)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(Monitor,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    144,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(32, 32, 32)
                        .addComponent(Fechar,  javax.swing.GroupLayout.PREFERRED_SIZE,
144, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel3)
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                    .addComponent(Fechar,      javax.swing.GroupLayout.PREFERRED_SIZE,
42, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(Monitor,    javax.swing.GroupLayout.PREFERRED_SIZE,
42, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
```

```
                new GarcomDigital().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton Fechar;
    private javax.swing.JButton Monitor;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    // End of variables declaration
}
```

Apêndice E-Código Config

```
package garcomdigital;

import java.awt.Dimension;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import gnu.io.*;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.Enumeration;
import java.util.HashMap;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;

public class Config extends javax.swing.JDialog {
    protected HashMap map = new HashMap();
    /** The name of the choice the user made. */
    protected String selectedPortName;
    protected Integer selectedBaudRate = 0;
    /** The CommPortIdentifier the user chose. */
    protected CommPortIdentifier selectedPortIdentifier;
    protected boolean errorFlag = false;
    protected boolean connected = false;
    public Config(JFrame parent) {
        super(parent,true);
        initComponents();
        pack();
        Rectangle parentBounds = parent.getBounds();
        Dimension size = getSize();
        // Center in the parent
        int x = Math.max(0, parentBounds.x + (parentBounds.width - size.width) / 2);
        int y = Math.max(0, parentBounds.y + (parentBounds.height - size.height) / 2);
        setLocation(new Point(x, y));
        populate();
        itemChoice();
        ButtonOK.setEnabled(false);
        ButtonOK.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ConnectActionPerformed(evt);
            }
        });
        ButtonCancel.addActionListener(new ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            closeButtonActionPerformed(evt);
        }
    });

}


private void closeButtonActionPerformed(java.awt.event.ActionEvent evt) {
    setVisible(false);
    dispose();
}
private void itemChoice() {
    COMList.addItemListener(new ItemListener() {

        public void itemStateChanged(ItemEvent e) {
            // Get the name
            selectedPortName = (String) ((JComboBox) e.getSource())
                .getSelectedItem();
            // Get the given CommPortIdentifier
            selectedPortIdentifier = (CommPortIdentifier) map.get(selectedPortName);
            if(selectedPortName != null && selectedBaudRate != null) {
                ButtonOK.setEnabled(true);
            }
        }
    });
    BaudList.addItemListener(new ItemListener() {

        public void itemStateChanged(ItemEvent e) {
            selectedBaudRate = (Integer) ((JComboBox) e.getSource())
                .getSelectedItem();
            if(selectedPortName != null && selectedBaudRate != null) {
                ButtonOK.setEnabled(true);
            }
        }
    });

}
private void ConnectActionPerformed(java.awt.event.ActionEvent evt) {
    setVisible(false);
    dispose();
}

public String getSelectedPort() {
    return selectedPortName;
}
public CommPortIdentifier getSelectedIdentifier() {
    return selectedPortIdentifier;
}
public int getBaudRate() {
    return selectedBaudRate;
```

```
    }
    public boolean hasError() {
        return errorFlag;
    }
    public boolean Connected() {
        return connected;
    }
    public void showDialog() {
        if(hasError()) {
            JOptionPane.showMessageDialog(null, "Nenhuma porta encontrada.\n\nVerifique a conexão com a porta USB e tente novamente.\n", "Erro", JOptionPane.ERROR_MESSAGE);
        }
        else {
            this.setVisible(true);
        }
    }
    private void populate() {

        Enumeration pList = CommPortIdentifier.getPortIdentifiers();

        while (pList.hasMoreElements()) {
          CommPortIdentifier cpi = (CommPortIdentifier) pList.nextElement();

          map.put(cpi.getName(), cpi);
          if (cpi.getPortType() == CommPortIdentifier.PORT_SERIAL) {
            COMList.addItem(cpi.getName());
          }
        }
        if(COMList.getItemCount() > 0) {
            BaudList.addItem((Integer)1200);
            BaudList.addItem((Integer)2400);
            BaudList.addItem((Integer)4800);
            BaudList.addItem((Integer)9600);
            COMList.setSelectedIndex(-1);
            BaudList.setSelectedIndex(-1);
        }
        else {
            ButtonCancel.setText("Fechar");
            errorFlag = true;
        }
    }

    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        ButtonCancel = new javax.swing.JButton();
        ButtonOK = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
```

```
        jLabel2 = new javax.swing.JLabel();
        COMList = new javax.swing.JComboBox();
        BaudList = new javax.swing.JComboBox();


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Porta Serial");

        ButtonCancel.setText("Fechar");

        ButtonOK.setText("Conectar");


jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFa
ctory.createEtchedBorder(), "Propriedades"));

        jLabel1.setText("Porta COM");

        jLabel2.setText("Velocidade (bps)");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(11, 11, 11)
                        .addComponent(jLabel1))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel2)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                    .addComponent(COMList, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(BaudList, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(41, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
```

```
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(COMList,  javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel1))
            .addGap(24, 24, 24)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(jLabel2)
                .addComponent(BaudList,  javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
      );

      javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
      getContentPane().setLayout(layout);
      layout.setHorizontalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                .addGroup(layout.createSequentialGroup()
                   .addGap(28, 28, 28)
                   .addComponent(ButtonCancel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                   .addComponent(ButtonOK))
                .addGroup(layout.createSequentialGroup()
                   .addContainerGap()
                   .addComponent(jPanel1,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
            .addContainerGap())
      );
      layout.setVerticalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addContainerGap()
```

```
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                .addComponent(ButtonOK)
                .addComponent(ButtonCancel))
            .addContainerGap())
        );

        pack();
    }// </editor-fold>

    // Variables declaration - do not modify
    private javax.swing.JComboBox BaudList;
    private javax.swing.JButton ButtonCancel;
    private javax.swing.JButton ButtonOK;
    private javax.swing.JComboBox COMList;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration
}
```

Apêndice F- Código Monitorar Mesas

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * Monitor.java
 *
 * Created on Oct 25, 2011, 6:40:59 PM
 */
package garcomdigital;
import java.io.*;
import java.util.*;
import gnu.io.*;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
/**
 *
 * @author Aline
 */
public class Monitor extends javax.swing.JDialog implements Runnable,
SerialPortEventListener {

    static Enumeration       portList;
    static CommPortIdentifier portId;
    static SerialPort        serialPort;
    static OutputStream      outputStream;
    static boolean           outputBufferEmptyFlag = false;
    static boolean          conectado = false;
    static InputStream       inputStream;
    Thread                   readThread;
    Integer                 mesa = 0;
    static boolean           solicitacao = false;
    boolean                 AtdM1 = false;
    boolean                 AtdM2 = false;
    int                 solicitado = 0;

    final                Runnable                Aviso                =                (Runnable)
Toolkit.getDefaultToolkit().getDesktopProperty("win.sound.exclamation");
    /** Creates new form Monitor */
    public Monitor(JFrame parent) {
        super(parent,true);
        initComponents();
        pack();
        setTitle("Monitor de mesas - por Aline Ataíde");
        ImprimeLog("Monitor Iniciado - "+ DataAtual());
        AcoesBotoes();
        Rectangle parentBounds = parent.getBounds();
        Dimension size = getSize();
        // Center in the parent
        int x = Math.max(0, parentBounds.x + (parentBounds.width - size.width) / 2);
        int y = Math.max(0, parentBounds.y + (parentBounds.height - size.height) / 2);
        setLocation(new Point(x, y));


        javax.swing.Timer t = new javax.swing.Timer(1000, new ClockListener());
        t.start();
        addWindowListener(new WindowEventHandler());
```

```
}


class ClockListener implements ActionListener {
   @Override
      public void actionPerformed(ActionEvent e) {
               //... Whenever this is called, get the current time and
               //    display it in the textfield.
      Calendar now = Calendar.getInstance();
      int h = now.get(Calendar.HOUR_OF_DAY);
      int m = now.get(Calendar.MINUTE);
      int s = now.get(Calendar.SECOND);
      //relogio.setText("" + h + ":" + m + ":" + s);
      relogio.setText(String.format("%1$tH:%1$tM:%1$tS", now));
      //... The following is an easier way to format the time,
      //    but requires knowing how to use the format method.
      //_timeField.setText(String.format("%1$tH:%1$tM:%1$tS", now));
      }
}

class WindowEventHandler extends WindowAdapter {
   @Override
   public void windowClosing(WindowEvent evt) {
      desconectar();
      dispose();
   }
}

public final void AcoesBotoes() {
   M1.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         M1Atendida(evt);
      }
   });
   M2.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         M2Atendida(evt);
      }
   });
   Parar.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         PararMonitoramento(evt);
      }
   });
```

```
    }

    private void M1Atendida(java.awt.event.ActionEvent evt) {
        if(AtdM1) {
            M1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png")));
            ImprimeLog("Solicitação atendida na mesa: 01");
            AtdM1 = false;
            if(solicitado > 0)
                solicitado--;
        }
    }

    private void M2Atendida(java.awt.event.ActionEvent evt) {
        if(AtdM2) {
            M2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png")));
            ImprimeLog("Solicitação atendida na mesa: 02");
            AtdM2 = false;
            if(solicitado > 0)
                solicitado--;
        }
    }

    private void PararMonitoramento(java.awt.event.ActionEvent evt) {
        desconectar();
        dispose();
    }
    public void conectar(CommPortIdentifier Id, Integer baudrate) {

        try {
            serialPort = (SerialPort) Id.open("serialComm",1000);

        } catch (PortInUseException e) {
            JOptionPane.showMessageDialog(null, "Não foi possível conectar.\nPorta em
uso "+Id.getName(), "Aviso", JOptionPane.WARNING_MESSAGE);

        }

        try {
                inputStream = serialPort.getInputStream();
            } catch (IOException e) {}

        try {
                serialPort.addEventListener(this);
            } catch (TooManyListenersException e) {}

            serialPort.notifyOnDataAvailable(true);
```

```
    try {
        serialPort.setSerialPortParams(baudrate,
                                SerialPort.DATABITS_8,
                                SerialPort.STOPBITS_1,
                                SerialPort.PARITY_NONE);
        outputStream = serialPort.getOutputStream();

        conectado = true;
    }
    catch (UnsupportedCommOperationException e) {}
    catch (IOException e) {}
    readThread = new Thread(this);

        readThread.start();
}

@Override
public void run() {
        try {
            Thread.sleep(20000);
        } catch (InterruptedException e) {}
}

@Override
public void serialEvent(SerialPortEvent event) {
        switch (event.getEventType()) {

        case SerialPortEvent.BI:

        case SerialPortEvent.OE:

        case SerialPortEvent.FE:

        case SerialPortEvent.PE:

        case SerialPortEvent.CD:

        case SerialPortEvent.CTS:

        case SerialPortEvent.DSR:

        case SerialPortEvent.RI:

        case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
            break;

        case SerialPortEvent.DATA_AVAILABLE:
        dataAvailable(event);
            break;
```

```
        }
    }

    protected void dataAvailable(SerialPortEvent event) {
        int dados = 0;
        try {
            dados = inputStream.read();
            if (dados == 254) {
                switch (mesa) {
                    case 72:
                        if(solicitado <= 0)
                            M1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323030974_
Circle_Red.png")));
                        else
                            M1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323030994_
Circle_Green.png")));
                        ImprimeLog("Solicitação da mesa: 01");
                        solicitado++;
                        if (Aviso != null)
                            Aviso.run();
                        //JOptionPane.showMessageDialog(null, "Garçom,\nCliente na mesa
01 solicitou atendimento!", "Atendimento",
JOptionPane.INFORMATION_MESSAGE);

                        AtdM1 = true;

                        break;
                    case 81:
                        if(solicitado <= 0)
                            M2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323030974_
Circle_Red.png")));
                        else
                            M2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323030994_
Circle_Green.png")));
                        ImprimeLog("Solicitação da mesa: 02");
                        solicitado++;
                        if (Aviso != null)
                            Aviso.run();
                        //JOptionPane.showMessageDialog(null, "Garçom,\nCliente na mesa
02 solicitou atendimento!", "Atendimento",
JOptionPane.INFORMATION_MESSAGE);

                        AtdM2 = true;

                        break;
```

```
                }
            }
            else {
                mesa = dados;
            }


        } catch(IOException ex) {}
}

public void desconectar() {
    if (conectado()) {
        try {
            serialPort.removeEventListener();
            outputStream.close();
            inputStream.close();
            serialPort.close();
            conectado = false;
        } catch (IOException e) {

        }
    }
}

public boolean conectado() {
    return conectado;
}

    private String TempoAtual() {
    DateFormat dateFormat = new SimpleDateFormat("[HH:mm:ss] - ");
    Date date = new Date();
    return dateFormat.format(date);
}
private String DataAtual() {
    DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
    Date date = new Date();
    return dateFormat.format(date);
}

private void ImprimeLog(String mensagem) {
    Log.append(TempoAtual() + mensagem + "\n");
    Log.setCaretPosition(Log.getText().length());
}

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jPanel1 = new javax.swing.JPanel();
    relogio = new javax.swing.JTextField();
```

```
        M1 = new javax.swing.JButton();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        M2 = new javax.swing.JButton();
        jTextField4 = new javax.swing.JTextField();
        M3 = new javax.swing.JButton();
        jTextField5 = new javax.swing.JTextField();
        M4 = new javax.swing.JButton();
        jTextField6 = new javax.swing.JTextField();
        M5 = new javax.swing.JButton();
        jTextField7 = new javax.swing.JTextField();
        M6 = new javax.swing.JButton();
        M7 = new javax.swing.JButton();
        jTextField8 = new javax.swing.JTextField();
        M8 = new javax.swing.JButton();
        jTextField9 = new javax.swing.JTextField();
        M9 = new javax.swing.JButton();
        jTextField10 = new javax.swing.JTextField();
        M10 = new javax.swing.JButton();
        jTextField11 = new javax.swing.JTextField();
        M11 = new javax.swing.JButton();
        jTextField12 = new javax.swing.JTextField();
        M12 = new javax.swing.JButton();
        jTextField13 = new javax.swing.JTextField();
        Parar = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        Log = new javax.swing.JTextArea();
        jLabel1 = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);


jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFa
ctory.createEtchedBorder(javax.swing.border.EtchedBorder.RAISED), "Mesas"));

        relogio.setEditable(false);
        relogio.setFont(new java.awt.Font("SansSerif", 1, 14));
        relogio.setText("00:00:00");
        relogio.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        M1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M1.setFocusPainted(false);
        M1.setVerifyInputWhenFocusTarget(false);

        jTextField2.setEditable(false);
        jTextField2.setFont(new java.awt.Font("Verdana", 1, 24));
```

```
        jTextField2.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField2.setText("01");

        jTextField3.setEditable(false);
        jTextField3.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField3.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField3.setText("02");

        M2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M2.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M2.setFocusPainted(false);
        M2.setVerifyInputWhenFocusTarget(false);

        jTextField4.setEditable(false);
        jTextField4.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField4.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField4.setText("03");

        M3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M3.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M3.setFocusPainted(false);
        M3.setVerifyInputWhenFocusTarget(false);

        jTextField5.setEditable(false);
        jTextField5.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField5.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField5.setText("04");

        M4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M4.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M4.setFocusPainted(false);
        M4.setVerifyInputWhenFocusTarget(false);

        jTextField6.setEditable(false);
        jTextField6.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField6.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField6.setText("05");

        M5.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M5.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M5.setFocusPainted(false);
        M5.setVerifyInputWhenFocusTarget(false);
```

```
        jTextField7.setEditable(false);
        jTextField7.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField7.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField7.setText("06");

        M6.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M6.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M6.setFocusPainted(false);
        M6.setVerifyInputWhenFocusTarget(false);

        M7.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M7.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M7.setFocusPainted(false);
        M7.setVerifyInputWhenFocusTarget(false);

        jTextField8.setEditable(false);
        jTextField8.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField8.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField8.setText("07");

        M8.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M8.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M8.setFocusPainted(false);
        M8.setVerifyInputWhenFocusTarget(false);

        jTextField9.setEditable(false);
        jTextField9.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField9.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField9.setText("08");

        M9.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M9.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M9.setFocusPainted(false);
        M9.setVerifyInputWhenFocusTarget(false);

        jTextField10.setEditable(false);
        jTextField10.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField10.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField10.setText("09");
```

```
        M10.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M10.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M10.setFocusPainted(false);
        M10.setVerifyInputWhenFocusTarget(false);

        jTextField11.setEditable(false);
        jTextField11.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField11.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField11.setText("10");

        M11.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M11.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M11.setFocusPainted(false);
        M11.setVerifyInputWhenFocusTarget(false);

        jTextField12.setEditable(false);
        jTextField12.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField12.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField12.setText("11");

        M12.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/garcomdigital/Imagens/1323031057_
Circle_Grey.png"))); // NOI18N
        M12.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        M12.setFocusPainted(false);
        M12.setVerifyInputWhenFocusTarget(false);

        jTextField13.setEditable(false);
        jTextField13.setFont(new java.awt.Font("Verdana", 1, 24));
        jTextField13.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField13.setText("12");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(M1, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
45, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(M2,     javax.swing.GroupLayout.PREFERRED_SIZE,     38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField3,  javax.swing.GroupLayout.PREFERRED_SIZE,
45, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(M3,     javax.swing.GroupLayout.PREFERRED_SIZE,     38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField4,  javax.swing.GroupLayout.PREFERRED_SIZE,
45, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
         .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(M4,     javax.swing.GroupLayout.PREFERRED_SIZE,     38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField5,  javax.swing.GroupLayout.PREFERRED_SIZE,
45, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(M5,     javax.swing.GroupLayout.PREFERRED_SIZE,     38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField6,  javax.swing.GroupLayout.PREFERRED_SIZE,
45, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(M6,     javax.swing.GroupLayout.PREFERRED_SIZE,     38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField7,  javax.swing.GroupLayout.PREFERRED_SIZE,
45, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
         .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(M7,     javax.swing.GroupLayout.PREFERRED_SIZE,     38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
            .addComponent(jTextField8,  javax.swing.GroupLayout.PREFERRED_SIZE,
45, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(M8,    javax.swing.GroupLayout.PREFERRED_SIZE,    38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField9,  javax.swing.GroupLayout.PREFERRED_SIZE,
45, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(M9,    javax.swing.GroupLayout.PREFERRED_SIZE,    38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField10,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    45,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
          .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(M10,    javax.swing.GroupLayout.PREFERRED_SIZE,    38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField11,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    45,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(M11,    javax.swing.GroupLayout.PREFERRED_SIZE,    38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField12,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    45,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(M12,    javax.swing.GroupLayout.PREFERRED_SIZE,    38,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField13,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    45,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
```

```
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
            .addContainerGap(250, Short.MAX_VALUE)
            .addComponent(relogio,        javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap())
    );

    jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL,          new
java.awt.Component[] {M1, jTextField2});

    jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL,          new
java.awt.Component[] {M2, jTextField3});

    jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL,          new
java.awt.Component[] {M3, M4, M5, M6, jTextField4, jTextField5, jTextField6,
jTextField7});

    jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL,          new
java.awt.Component[] {M10, M11, M12, M7, M8, M9, jTextField10, jTextField11,
jTextField12, jTextField13, jTextField8, jTextField9});

    jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
            .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.T
RAILING, false)
                .addComponent(jTextField2,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE,                                41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M1,    javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addComponent(jTextField4,
javax.swing.GroupLayout.PREFERRED_SIZE,                                41,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(M3)
            .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE,                                41,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(M2))
```

```
            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                .addComponent(jTextField5,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M4)
                .addComponent(jTextField6,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M5)
                .addComponent(jTextField7,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M6))
            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                .addComponent(jTextField8,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M7)
                .addComponent(jTextField9,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M8)
                .addComponent(jTextField10,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M9))
            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                .addComponent(jTextField11,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M10)
                .addComponent(jTextField12,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M11)
                .addComponent(jTextField13,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(M12))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
            .addComponent(relogio, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(74, 74, 74))
      );

      jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL, new
java.awt.Component[] {M1, jTextField2});

      jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL, new
java.awt.Component[] {M2, jTextField3});

      jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL, new
java.awt.Component[] {M3, M4, M5, M6, jTextField4, jTextField5, jTextField6,
jTextField7});

      jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL, new
java.awt.Component[] {M10, M11, M12, M7, M8, M9, jTextField10, jTextField11,
jTextField12, jTextField13, jTextField8, jTextField9});

      Parar.setText("Parar Garçom Digital");

      Log.setColumns(20);
      Log.setEditable(false);
      Log.setRows(5);

Log.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border.Etch
edBorder.RAISED));
      Log.setDisabledTextColor(new java.awt.Color(0, 0, 0));
      Log.setEnabled(false);
      jScrollPane1.setViewportView(Log);

      jLabel1.setText("OBS: Para marcar como atendido, clique na lâmpada acesa.");

      javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
      getContentPane().setLayout(layout);
      layout.setHorizontalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(layout.createSequentialGroup()
            .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
               .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
332, Short.MAX_VALUE)
               .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
               .addComponent(Parar, javax.swing.GroupLayout.Alignment.TRAILING)
               .addComponent(jLabel1))
```

```
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 293,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(Parar)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel1)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    // Variables declaration - do not modify
    private javax.swing.JTextArea Log;
    private javax.swing.JButton M1;
    private javax.swing.JButton M10;
    private javax.swing.JButton M11;
    private javax.swing.JButton M12;
    private javax.swing.JButton M2;
    private javax.swing.JButton M3;
    private javax.swing.JButton M4;
    private javax.swing.JButton M5;
    private javax.swing.JButton M6;
    private javax.swing.JButton M7;
    private javax.swing.JButton M8;
    private javax.swing.JButton M9;
    private javax.swing.JButton Parar;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField jTextField10;
    private javax.swing.JTextField jTextField11;
    private javax.swing.JTextField jTextField12;
    private javax.swing.JTextField jTextField13;
    private javax.swing.JTextField jTextField2;
```

```
private javax.swing.JTextField jTextField3;
private javax.swing.JTextField jTextField4;
private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
private javax.swing.JTextField jTextField7;
private javax.swing.JTextField jTextField8;
private javax.swing.JTextField jTextField9;
private javax.swing.JTextField relogio;
// End of variables declaration
```